文章编号: 1001-2486 (2000) 04-0057-04

动态的主动复制协议的研究与设计*

史殿习, 吴泉源, 王怀民, 邹鹏

(国防科技大学计算机学院, 湖南 长沙 410073)

摘 要: 从软件容错的角度出发, 将组通信技术引入到主动复制技术中, 以组通信中的虚拟同步机制为基础, 提出一个新的基于协调者的主动复制协议。该协议保证, 当协调者进程发生失效时, 能够动态地选择一个新的协调者, 并保证所有副本进程的状态是一致的; 且该协议具有响应时间短、效率高等优点。

关键词: 容错; 复制; 组通信; 虚拟同步

中图分类号: TP391 文献标识码: A

The Research and Design of the Dynamic Active Replication Fault-Tolerant Protocol

SHI Dian-xi, WU Quan-yuan, WANG Huai-min, ZHOU Peng

(College of Computer, National Univ. of Defense Technology, Changsha 410073, China)

Abstract: From the software fault-tolerance view, we introduce the group communication technology to the active replication technology, propose a new dynamically active replication protocol based on a coordinator. When the coordinator is crashed, the protocol can dynamically select a new coordinator process and ensure the state of all the backup service processes are consistent. The advantage of the protocol is the short response time and high efficiency.

Key words: fault-tolerance; replication; group communication; virtual synchronization

在主动复制[1]技术中, 要保证所有副本的状态是一致的, 必须要求所有的副本按相同的顺序处理所有来自不同客户的请求。因此, 设计一个主动复制协议就演变成如何设计一个全序 (Total Order) 协议。为了满足分布式应用对可靠性的要求, 提高系统的容错能力, 我们提出了一个新的基于协调者的全序算法, 该算法的优点是效率高, 在协调者失效时, 能够动态地选择一个新的协调者, 而且保证所有副本状态是一致的; 并以该算法为基础, 我们设计实现了一个动态的主动复制协议, 支持分布式应用的开发。

1 相关工作

目前, 人们已经设计出很多的全序算法。然而, 文献 [2] 中证明, 在一个异步系统中, 在成员可能发生失效的情况下, 不存在一个全序的实现算法; 为了解决这一问题, 文献 [2] 中对异步系统进行扩展, 在异步系统中引入了不可靠的失效检测器的概念, 进而提出了一个全序算法, 并证明该算法是正确的。该算法适合于一次对多条消息进行排序, 而不适合一次对一条消息进行排序, 其原因在于决定一条消息的顺序需要多个回合。在对现有的全序算法研究的基础上, 基于组通信中的虚拟同步机制[3], 提出了一个基于协调者的全序算法。

2 系统模型

在基于复制的软件容错方法中, 可以将分布在系统中不同节点上的副本看成一个副本组, 客户与副本之间的交互可以看成客户与一个副本组之间的交互。因此, 我们将组通信技术引入到基于复制的软件容错方法中, 提出了一个通用的容错模型, 如图 1 所示。假设系统是一个异步的分布式系统, 即系统中对进程之间的消息传递延时没有限制, 对进程的执行速度没有限制, 且系统中没有一个同步时

* 收稿日期: 2000-02-27
基金项目: 国家 863 计划资助项目 (863-306-ZD-04)
作者简介: 史殿习 (1966-), 男, 博士生。

钟或一个全局时钟。

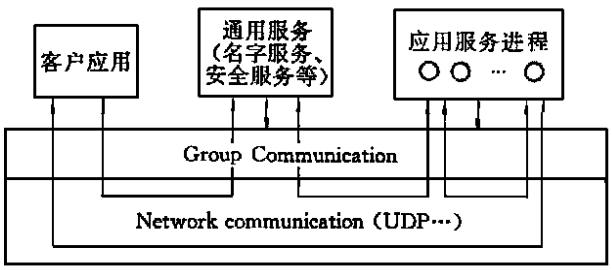

图 1 基于组通信的容错模型

Fig. 1 The fault-tolerant model based on the group communication

在该模型中，网络通信层为客户与服务进程之间、服务进程与服务之间提供异步的点到点通讯通道，它保证点到点的通信是可靠的，并且消息的顺序是 FIFO 的；组通信层为副本组中成员之间的组播通信、故障处理及组成员之间的同步处理提供有效的支持机制，这些机制包括：可靠的组播通信原语 ReliabkeMcast ()、虚拟同步过程 VSC_Mcast ()（见 3.2 节）等；通用服务主要包括名字服务、安全服务等分布式系统中公用的服务，其主要作用是维护系统中副本组，为应用查询、定位系统中的服务提供支持。副本组在系统初始化时向名字服务器注册，名字服务器中设有一个监测器，对注册的组成员进行监测，当有成员发生失效时，通知组内其它成员。副本组成员在完成注册后，一直处于激活状态，当接收到服务请求时，立刻对请求进行处理，处理完成之后将结果返回。

3 虚拟同步

设计一个基于协调者的全序算法的关键点在于，当协调者失效时，如何选择一个唯一的新协调者，以及如何保持所有的副本的状态是一致的。为了满足这两个条件，我们将组通信中的虚拟同步机制[3]引入到全序算法中。

3.1 虚拟同步的定义

虚拟同步机制与组的成员关系变化是密切相关的。为了表示上的方便，我们引入视图的概念，视图是对组成员关系的一种抽象。

定义 1（视图） 假设一个组 g，组 g 的一个视图是指，由组 g 中处于正确的运行状态且彼此之间可以进行相互通信的成员进程组成的进程子集，而且该子集被其中的所有成员进程都认同。我们用 $view_i$ (g) 来表示组 g 的一个视图，每一个视图都有唯一的标识。

在一个组中，组成员之间的组播通信随时受组视图变化的影响，对于一个组成员来说，不仅要处理成员之间的组播消息，而且还要处理视图变化通知消息，并且这两类消息相互交织穿插在成员所处理的消息流中。虚拟同步机制的作用就是对这两类消息进行协调管理，在组视图发生变化的情况下，保证正确的组成员之间对这两类消息的同步处理。下面我们给出虚拟同步的严格定义。

定义 2（虚拟同步） 假设组 g，视图 $view_x$ (g) 和 $view_y$ (g)，且 $view_y$ (g) 是 $view_x$ (g) 的直接后继，我们称组 g 中的通信是虚拟同步的，当且仅当如果存在一个进程 $p \in view_x$ (g)，在视图 $view_x$ (g) 中已经传递（处理）了消息 m，且已安装了下一个视图 $view_y$ (g)，则所有已经安装下一个视图 $view_y$ (g) 的进程在安装 $view_y$ (g) 之前已经传递（处理）了消息 m。

3.2 虚拟同步算法

为了在组成员之间实现虚拟同步通信，我们对文献 [2] 中的 ◇S-Consensus 算法（详见文献 [2]）进行修改（将其标记为 ◇SM-Consensus），以 ◇SM-Consensus 为基础，我们设计了一个虚拟同步算法 VSC_Mcast，其描述如下所示。当组的视图发生变化（如成员失效、新的成员加入等）时，当前视图中所有正确的成员就启动算法 VSC_Mcast，以便所有正确的成员在下一个视图和在当前视图中所传递

的消息集合达成一致。

Procedure VSC_Mcast ()

(1) 每个成员 p 计算下一个视图的估计值 $estiview_p$ 及计算在当前视图中所传递的消息集合 $dlvrmsgset_p$ (m), 并将这两个估计值构造成一个新的估计值 $estimate_p \leftarrow \{ estiview_p, dlvrmsgset_p (m) \}$;

(2) 以 $estimate_p$ 为参数调用修改后的 ◇SM-Consensus 算法, 即调用过程 ConsistPro ()。该过程的作用是使所有的正确成员在下一个视图上和当前视图中所传递的消息集合上达成一致; result ← ConsistPro ($estimate_p$);

(3) return result

算法 ◇SM-Consensus (即过程 ConsistPro ()) 与算法 ◇S-Consensus 的不同之处在于, ◇SM-Consensus 算法对 ◇S-Consensus 算法中的第二阶段进行了修改, 即协调者 p_c 根据接收到的各个成员的提议消息来计算新的估计值 $estimate_{pc}$的, p_c 关于下一个视图的估计值 $NextView_{pc}$是所有属于当前估计视图中的成员的 $NextView_q$ 的交集, 这样可以保证下一个估计视图中不包含所有被怀疑的成员; $DlvMsgSet_{pc}$ (m) 是所有属于当前估计视图中的成员的 $DlvMsgSet_{pi}$ (m) 的并集, 这样可以保证属于下一个估计视图中的每一个成员不遗漏一条在当前视图中被其它成员传递消息。

我们所提出的这个虚拟同步算法的优点在于, 在参与同步的成员的失效个数不超过一半的情况下, 该协议保证当前视图中所有正确的成员在下一个视图和在当前视图中所传递的消息集合达成一致, 并且满足终止性、一致性及有效性等特性; 而其它的虚拟同步协议在特定的情况下, 并不能完全满足终止性、一致性及有效性等特性, 即不能完全满足虚拟同步的语义。

4 动态的主动复制协议

在对虚拟同步机制研究的基础上, 我们以虚拟同步算法 VSC_Mcast 为基础, 提出了一个新的全序算法 TOMmcast, 并以该算法为基础, 设计了一个新的主动复制协议 ActivePro, 该协议由客户处理协议 ActiveCliPro 和副本成员处理协议 ActiveMembPro 两部分组成 (我们假定系统中绝大多数副本是正确的)。其中, 成员处理协议主要由 TOMmcast 算法构成。客户协议的主要功能是, 通过可靠的组播通信原语 ReliableMcast (), 将请求发送给副本组 g, 并等待来自绝大多数副本的应答消息, 最后将结果返回给客户; 当接收到来自成员关系服务的通知消息时, 则更新当前的组视图。

副本成员协议 ActiveMembPro 的主要功能是, 利用一个协调者来对发送给副本组中的组播消息进行排序, 该协议保证所有正确的副本进程按相同的顺序传递所有的消息。其算法描述如下所示:

(1) 当一个副本接收到来自客户的请求 req_c^j 时, 将 req_c^j 保存到消息缓冲区中, 直到从协调者那里接收到该消息的序号再决定是否将请求 req_c^j 传递给应用 (即对请求 req_c^j 进行处理);

(2) 当协调者接收到一条请求 req_c^j 时, 给请求 req_c^j 分配一个正确的序号 $order_{coord}$ ($order_{coord}$是一个递增函数), 并将消息< ID (req_c^j), $order_{coord}$> 发送给当前视图中的其它副本成员;

(3) 当一个副本接收到来自协调者的关于请求 req_c^j 的序号信息< ID (req_c^j), $order_{coord}$> 时, 判断该序号是否是要传递的下一条信息的序号 (在请求 req_c^j 之前, 该成员所传递的最后一条消息的序号是 $order_{coord}-1$), 如果是, 则将请求 req_c^j 传递给应用 (对请求进行处理), 将处理结果返回给客户; 并将该序号消息< ID (req_c^j), $order_{coord}$> 保存到消息队列 $DlvMsgSet_p^i$ 中 (为了避免消息的益处, 我们设计了专门的协议来对队列中的消息进行回收);;

(4) 当副本成员接收到一条来自成员关系服务器的关于协调者失效的通知消息时, 则副本成员调用虚拟同步处理过程 VSC_Mcast () 进行同步处理, VSC_Mcast () 返回的结果为 $\{ V_{i+1} (g), DlvMsgSet^i \}$, 其中, $V_{i+1} (g)$ 表示新的组视图, $DlvMsgSet^i$ 表示副本进程在视图 $V_i (g)$ 中所要传递的消息集合;

(5) 同步过程处理完成之后, 所有副本成员在当前视图内所传递的消息集合上达成一致, 并在新

的视图上达成一致，每个副本成员判断其在当前视图中是否有尚未传递的消息，如果有尚未传递的消息集合，则按所有副本成员在 VSC_Mcast () 过程中所达成的顺序将这些消息进行处理，而后每个副本使用特定的策略选定一个新的副本作为协调者来对所收到的消息进行排序；

在 ActivePro 协议中，协调者使用 ReliableMcast () 通信原语将序号消息可靠地发送给其它副本服务进程，由于 ReliableMcast () 是可靠的且保证 FIFO 顺序的组播通信原语，因此，我们可以证明该协议所有副本服务进程都按相同的顺序处理所有的请求；同时，当协调者进程发生失效时，由于有虚拟同步机制的支持，我们可以证明所有副本服务进程在当前的视图内传递相同的消息集合；因此，无论协调者进程失效与否，上述协议始终保证所有副本服务进程的状态是一致的。因而，该协议是正确的(因篇幅所限，在此我们不作详细证明)。

5 结束语

与现有的主要的主动复制协议[1]相比，由于 ActivePro 协议中利用一个协调者来对客户请求进行排序，因此，在正常情况下，决定一条请求的顺序只需一个回合，而文献 [2] 中的协议需要多个回合；当协调者发生失效时，对尚未排序的消息进行统一排序，从而大大提高系统的效率，因此，ActivePro 协议响应时间短，效率高。

在对上述协议研究的基础上，为了支持军事指挥系统的开发，我们在自己开发的组通信开发工具 GCS[4]之上，设计实现了一个服务支持系统 FTS，该系统实现了我们上面所提出的算法，并且已经验证该算法是正确的。目前，我们正在将 FTS 应用到分布式数据库应用中。

参考文献:

[1] Schneider F. Distributed Systems [M], chapter 7: Replication Management using the State-Machine Approach: 169-197. Addison-Wesley, 2nd edition, 1993.

[2] Chandra T D, Toueg S. Unreliable Failure Detector for Reliable Distributed Systems [J]. Journal of the ACM, 1996, 43 (1): 225-267.

[3] Birman K, Joseph T. Reliable Communication in the Presence of Failures [J]. ACM Transactions on Commputer Systems, 1987, 5 (1): 47-76.

[4] 史殿习，吴泉源等. 协同应用中组通信服务的研究与设计 [J]. 计算机辅助设计与图形学报，1999, 12 (1): 77-80.