动态的被动复制容错协议的研究与设计

The Research and Design of the Dynamic Passive Fault-Tolerant Protocol

唐文胜　史殿习

（湖南师范大学计算机教学部　长沙410006）（国防科大计算机学院　长沙410073）

Abstract　In order to satisfy the reliable requirment of the distributed application such as distributed database and bank management applicatons, and to support the development of these applications, in order to overcome the limitation of the exiting protocol, we introduce the group communiction technology to the passive replication technology, propose a new dynamically passive replication protocol based on the virtual synchronization. When the primary crashes, the protocol can dynamically select a new primary service process and assure the state of all the backup service processes is consistent.

Keywords　Fault-tolerance, Repication, Group communication, Virtual synchronization

1　引言

随着应用需求的发展，分布式应用如分布式数据库应用、银行管理系统等对系统的可靠性要求越来越高。提高系统可靠性的一种主要技术是基于复制的软件容错技术，主要有两种方法：即被动复制方法（也称为主备份复制方法）[1]和主动复制方法[2]。与主动复制方法相比，被动复制方法需要较少的系统资源，且由于客户只与主副本进行交互，控制简单；因此，被动复制技术在实际应用中被广泛地使用。

现有的被动复制协议主要有文[1]中设计的一个基于同步时钟的协议，该协议假设系统中有一个同步时钟，并且利用同步时钟来保证主副本与后备副本之间的同步。然而，由于在分布式系统通常是不存在一个统一时钟的，因此，该协议具有极大的局限性。为了克服这种局限性，支持分布式应用的开发，我们将组通信技术引入到基于复制的软件容错方法中，以组通信中的虚拟同步机制为基础，提出一个新的动态的被动复制算法。该算法保证，当主副本进程发生失效时，能够动态地选择一个新的主副本，并保证所有后备副本状态是一致，并且具有良好的透明性。

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，主服务进程与后备服务进程之间的交互可以看成是主服务进程与其它组成员之间的交互。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。该模型假设系统是一个异步的分布式系统，且系统中没有一个同步时钟或一个全局时钟。

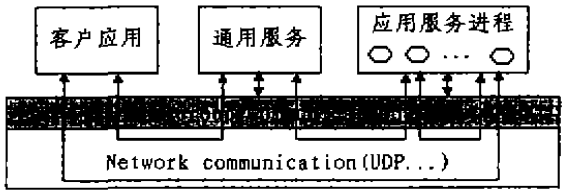

图1　基于组通信的容错模型

在该模型中，组通信层为副本组内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RMcast()、虚拟同步过程 VSC_Mcast()（见第3节）等；通用服务主要包括名字服务等分布式系统中公用的服务，其主要作用是维护和管理系统中的副本组，为查询、定位系统中的服务提供支持。

3　虚拟同步

在被动复制方法中，由于客户不直接与后备服务进程进行交互，因此，当后备服务进程失效时并不影响

唐文胜　硕士，讲师，主要研究方向：软件工程，分布式系统，多媒体，CAI等。史殿习　博士，主要研究方向：分布式系统，软件工程等。

系统的正常运行；但当主服务进程发生失效时，必须从后备服务进程中选择一个服务进程来承担主服务进程的角色，以便在主服务进程发生失效时，能够：(1)定义一个新的、唯一的服务进程来充当主服务进程的角色；而且，(2)新的主服务进程的状态与其它所有后备服务进程的状态是一致的。为了满足上述两个条件，我们将组通信中的虚拟同步机制引入到被动复制方法中。

3.1 虚拟同步的定义

虚拟同步机制[3]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，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

一致性问题[4]是对实现具有容错能力的分布式系统的一个最基本的抽象。一致性问题是在一个进程集合上定义的。初始时，每个进程都有机会提议一个值 v，该集合中的所有正确的进程最终必须在所提议的某个值上达成共识。为了有效地解决分布式系统中的一致性问题，Chandra 和 Toueg 对分布式异步系统进行了扩展，在分布式异步系统中引入了不可靠的失效检测器[4]的概念，在假定大多数进程是正确的条件下，Chandra 和 Toueg 提出了一个使用◇S 类失效检测器解决一致性问题的一致性算法[4](我们用◇S-Consensus 来标识该算法)，并证明◇S 类失效检测器是异步系统中解决一致性问题的最弱的类。

为了在组成员之间实现虚拟同步通信，我们将虚拟同步问题转化为一致性问题，并对◇S-Consensus 算法(详见文[4])，进行修改(将其标记为◇SM-Consensus)，以满足虚拟同步语义的要求。以◇SM-Consensus 为基础，我们设计了一个虚拟同步算法 VSC_Mcast，其描述如图2所示。当组的视图发生变化时(如成员失效、新的成员加入等)，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 ← (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←Consist Pro(estimate_p);
  (3)return result
```

图2 虚拟同步算法

VSV_Mcast 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，该算法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。

4 动态的被动复制协议

以组通信的支持机制为基础，我们提出了一个动态的被动复制协议 DPP，该协议以虚拟同步为基础，能够在主服务进程发生失效的情况下，动态地从后备服务进程中选择一个新的服务进程，并且保证所有的服务进程的执行状态是一致的。

为了算法表达上的方便，我们引入如下的标记：req_c^i 表示由客户 c 发送的第 i^{th} 请求；upd_p^i 表示由主服务进程产生的第 i^{th} 更新消息；res_p^i 表示由服务进程产生的给客户的第 i^{th} 应答消息；$state_p^i$ 表示后备服务进程处理完更新消息后的状态；$handle_1(req_c^i, state_p^{i-1}) \rightarrow (upd_p^i, state_p^i, res_p^i)$ 表示将客户 c 发送的第 i^{th} 请求转化为一条更新消息；$apply_1(udp_p^i, state_p^{i-1}) \rightarrow state_p^i$ 表示后备服务进程由于执行更新消息 udp_p^i，而将当前的状态 $state_p^{i-1}$ 修改为新的状态 $state_p^i$。

图3给出了以事件触发为中心的动态的被动复制协议的描述。副本组中每个成员 p 都执行如图3的代码。在初始化阶段，首先，每个成员 p 通过副本组的名字从名字服务器中获取副本 g 的当前视图 $V_i(g)$，该视图包含了副本组中当前所有正确的成员的引用信息；其次，每个成员 p 使用一个确定的函数 selectPrima($V_i(g)$)，从当前的视图 $V_i(g)$ 中选出一个主服务进程；最后，每个成员 p 将消息缓冲队列 $UpdMsgSet_p^i$ 置为空，队列 $UpdMsgSet_p^i$ 的作用是存储服务进程在当

前视图中所传递(处理)的消息，当主服务进程发生失效时，后备服务进程在虚拟同步处理过程中对队列 $UpdMsgSet_p$ 中的消息进行交换、比较，以便所有服务进程在当前视图 $V_i(g)$ 中所传递的消息集合上达成一致。

初始化完成之后，每个服务进程都处于等待状态，对主服务进程来说，当其接收到来自客户的服务请求 req_c^j 时，则对该请求进行处理，形成一条状态更新消息 $(upd_p^j, state_p^j, res_c^j)$，而后通过原语 RMcast() 将该更新消息组播给当前视图中所有的后备服务进程，并等待来自后备服务进程的应答消息 ACK_c^j；对后备服务进程来说，当接收到来自主服务进程的更新消息 $(upd_p^j, state_p^j, res_c^j)$ 时，则对其当前的状态进行更新，并给主服务进程返回一个应答消息 ACK_c^j；当其接收到来自失效监测器有关主服务进程失效的消息时，调用虚拟同步过程 VSC_Mcast() 进行同步处理，VSC_Mcast() 返回的结果为 $\{V_{i+1}(g), UpdMsgSet'\}$，其中，$V_{i+1}(g)$ 表示新的组视图，$UpdMsgSet'$ 表示服务进程在视图 $V_i(g)$ 中所要传递的消息集合；同步处理返回后，每个后备服务进程 p 从新的组视图中 $V_{i+1}(g)$ 选择一个新的主服务进程，并将该新服务进程重新注册到名字服务器中，以便由名字服务器来通知相关的客户应用和其它服务；而后对尚未处理的消息进行处理，处理完之后，更新其当前的视图。

```
Algorithm DPP
  UpdMsgSet_a^j ← φ;
  V_i(g) ← GetView(g);
  Prima ← SelectPrima(V_i(g));
  AckNum_c^j ← 0;
  While True do {
    — upon recv(req_c^j):
      (upd_p^j, state_p^j, res_c^j) ← handle(req_c^j, state_p^{j-1});
      RMcast(<upd_p^j, res_c^j>, V_i(g));
    — upon recv(Ack_reqc):
      AckNum_c^j ← AckNum_c^j + 1;
      If AckNum_c^j > ⌈(|g|+1)/2⌉ then
        AckNum_c^j ← 0;
        Send(res_c^j) to client c;
    — upon recv(<upd_p^j, res_c^j>):
      state_a^j ← apply(upd_p^j, state_a^{j-1});
      UpdMsgSet_p^j ← UpdMsgSet_p^j ∪ {<upd_p^j, res_c^j>};
      Send(Ack_c^j) to Prima;
    — upon recv(suspect(Prima)):
      { V_{i+1}(g), UpdMsgSet' } ← VSC_Mcast();
      Prima ← SelectPrima(V_{i+1}(g));
      UnDlvMsg_p^j ← UpdMsgSet' − UpdMsgSet_p^j;
      While UnDlvMsg_p^j ≠ φ do {
        < upd_p^j, res_c^j > ← select a msg from UnDlvMsg_p^j;
        state_p^j ← apply(upd_p^j, state_p^{j-1});
        if p = Prima then Send(res_c^j) to Client c;
      }endwhile
      V_i(g) ← V_{i+1}(g);
  }endwhile
```

图3 副本成员处理协议

在 DPP 协议中，由于客户与主副本之间的通信是通过可靠的点到点的单播通信，主服务进程在接收到来自客户的请求并进行处理后，使用 RMcast() 通信原语将更新消息可靠地发送给其它后备服务进程，由于 RMcast() 是可靠的保证且保证 FIFO 顺序的组播通信原语，因此我们可以证明协议所有后备服务进程都按相同的顺序处理所有的请求；同时，当主服务进程发生失效时，由于有虚拟同步机制的支持，可以证明所有后备服务进程好处在当前的视图内传递相同的消息集合；因此，无论主服务进程失效与否，上述协议始终保证所有后备服务进程的状态是一致的，因而，该协议是正确的。

结束语 与协议[1]相比，DPP 协议利用虚拟同步机制而不是利用同步时钟来实现成员之间的同步的，这对协议[1]是一个极大的改进。在对 DPP 协议研究的基础上，我们在组通信开发工具 GCS[5] 之上，设计实现了一个容错服务支持系统 FTS，该系统实现了上面所提出的算法，并且我们已经验证该算法是正确的。

参 考 文 献

1 Budhiraja N, et al. Distributed System, chapter 8: The Primary-Backup, Addison-Wesley, 2nd edition, 1993. 199～216

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

3 Birman K, Joseph T. Reliable Communication in the Presence of Failures. ACM Transactions on Computer Systems, 1987, 5(1): 47～76

4 Chandra T D, Toueg S. Unreliable Failure Detector for Reliable Distributed Systems. Journal of the ACM, 1996, 43(1): 225～267

5 史殿习，吴泉源，等. 协同应用中组通信服务的研究与设计. 计算机辅助设计与图形学学报，2000，12(1)