具有并发类库的C++*

杨延中 王 为 田籁声

(吉林大学计算机科学系 长春 130023)

摘要 本文探讨如何通过类库将并发性引入顺序面向对象语言,以C++为例,在并发类库中提供并发类及相应工具,使之支持分布并行的面向对象程序设计.本文介绍并发类库及语言底层支撑系统的设计与实现,最后给出初步测试结果.

关键词 并发面向对象,并发对象,运行支撑系统.

中图法分类号 TP311

并发面向对象技术是一个比较新的研究领域,许多问题有待解决.近年来,有许多面向对象并发编程的方案,可归纳出把并发性引入面向对象系统的3种不同方法:[1](1) 设计全新的并发面向对象语言;(2) 扩展现存的顺序面向对象语言;(3) 使用现有的顺序面向对象语言,通过外部库提供并发抽象.

早期研制的并发面向对象语言多数是新语言.当面向对象思想成熟和顺序面向对象语言开始普及后,提出了许多扩展现有顺序语言的方案.库的方法较新,并且受到早期工作的影响.它使用现有的顺序面向对象语言,并通过外部库来提供并发抽象,如PARC++[2],ACT++.[3]这种方法的优点是不需要修改原有顺序面向对象语言,保留原有编程方法,可以更好地利用面向对象技术中的一些特性,用户容易接受.但是,这种方法大多需要底层特殊软件(运行支撑系统)的支持,可移植性较差.若采用通用操作系统,将复杂性全部隐藏在类库中,实现起来较为困难.

本文以广泛使用的C++语言为基础,通过类库开发其并行性.语言运行支撑系统建立在通用操作系统之上.

1 并发模型

1.1 并发对象

本文可并发执行的基本单位是对象,称作并发对象.并发对象是进程与对象这两个概念的结合.与普通对象不同的是,并发对象具有自己的控制线索和独立的地址空间.多个并发对象可以并发运行,通过方法请求相互作用.发出请求的对象称为顾客,接收请求的称为服务员.一个并发对象可以既是服务员又是顾客.通常,服务员对象用于管理共享资源,内部封装了共享资源及对其操作的一组过程,类似于一个管程.

每个并发对象都是CONCURRENCY类派生类的一个实例.每个并发对象必须定义一个方法scheduler,用来说明并发对象的活动.并发对象可以在执行过程中动态创建,一个并发对象可以要求创建另一个并发对象.创建一个并发对象需要调用create方法,以派生新的进程执行相应的scheduler方法.

为充分满足松散耦合分布式环境下面向对象编程的要求,允许在远程机上创建并发对象,并且创建形式与本地创建相同,具有远程透明性.访问一个远程对象,要通过它在本地的代理发出请求和接受结果.代理对象与普通对象相同,可以单独生成.代理对象通过执行attach方法与实际对象建立联系,即获得实际对象的通信地址.attach方法支持多个不同作业共享一个对象.

1.2 远程方法请求,同步和互斥

在顺序面向对象语言中,方法请求是一个同步调用过程.为充分发挥对象间的并行性,CONCURRENCY类实现了一个并发对象间的非阻塞、异步方法请求机制.请求对象不需等待即可继续执行,需要结果时再等待,是一种基于异步报文传递的数据驱动的同步策略如图1所示.其中request方法发送远程请求,同时返回请求号claim_number,然后继续执行,需要结果时,由result方法根据请求号返回相应结果.所有请求和结果都作为实际报文通过透明的进程

* 本文研究得到国家自然科学基金资助.
作者杨延中,1968年生,硕士生,主要研究领域为计算机网络与分布式系统.王为,1972年生,硕士生,主要研究领域为计算机网络与分布式系统.田籁声,女,1938年生,教授,主要研究领域为计算机网络与分布式系统.

本文通讯联系人:杨延中,长春 130023,吉林大学计算机科学系
本文1996-12-24收到原稿,1997-06-13收到修改稿

间通信机制 IPC 实现. 另外,CONCURRENCY 类还提供一个方法 result_ready,用于非阻塞测试结果是否返回.

```
int claim_number;
T result_value; // T 为实际返回结果类型
claim_number=obj. request(method);
      ⋮      }与 obj 并发执行
return_value=obj. result(claim_number);
```

图 1 远程请求

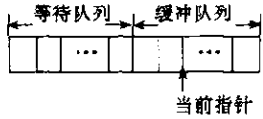

图2

每个服务员对象有一个方法请求队列 request_queue,如图 2 所示. 它由等待队列和缓冲队列构成,用于存放暂时不能处理和未来得及处理的请求. 服务员对象通过执行 get_request 方法获得请求,如果请求队列非空,则从请求队列中取,否则,等待. 为解决并发对象的同步和互斥问题,CONCURRENCY 类提供一个基本方法 put_request( ),当请求不被满足时放回等待队列. put_request 相当于管程中的 wait 原语,使发出请求的对象等待. 唤醒操作由方法 get_request 和 send_result 共同完成. get_request 负责选择等待的请求,send_result 发回结果,使等待对象继续执行. 在只使用 get_request 和 put_request 的情况下,服务员对象满足相同请求先到达先处理原则,并且不会出现忙式等待现象.

由于有了方法请求队列,自然使得并行请求串行化. 另外,对象内部的多个方法串行执行,在某种程度上简化了有关并发对象间的同步和互斥问题. 例如,有一个信号灯类

```
class Semaphor:CONCURRENCY{
    private:int s;
    public:p( ){s--;
        if(s>=0) send_result( );
        else 把当前请求放回请求队列等待;
        }
        v( ){s++;send_result( ); }
        ...
};
```

此时可不必考虑 p( )操作的不可中断性.

并发对象间的远程方法请求还有一种更透明的形式,通过类库中的 QUEUE 类实现. QUEUE 类是 CONCURRENCY 类的一个派生类,利用 QUEUE 类,可以使并发对象间的方法请求完全对用户透明,不需要使用 request,get_request,result 等显式调用. QUEUE 类对象的使用方法和普通队列完全一样,可以当作本地生成的普通顺序对象. 例如:有两个并发对象 a,b. a 读入一个数,b 把此数加 1.

```
#include "CONCURRENCY. h"
Class A:public CONCURRENCY{
private:QUEUE buf;
public:A(QUEUE q){buf=q;create();}
       void scheduler( )
       {int i;
        cin>>i;
        buf. put(i);
       }
};
main( )
{QUEUE buf;A a(buf);B b(buf);}
```

```
Class B:public CONCURRENCY{
private:QUEUE buf;
public:B(QUEUE q){buf=q;create();}
       void scheduler( )
       {int i=buf. get( );
        i=i+1;
       }
};
```

每个 QUEUE 类对象都是一个并发对象,相当于一个松散耦合分布式环境中的共享缓冲区. QUEUE 可以看作是请求队列或结果队列的一种简化,put 相当于发送请求,只是省略了方法名,get 相当于接收请求,队列为空时等待. 当接收方可以接收多种请求,put 的参数变为方法名+请求参数时,QUEUE 就成为真正的请求队列. 在实际应用中,为限制所占空间,可以规定队列的最大长度,当队列满时,put 等待. 当队列长度为 0,put 和 get 变成同步操作.

2 并发类库

并发类库中主要包括 CONCURRENCY 类、QUEUE 类、信号灯类、ANY 类和 REQUEST 类等. 其中最主要的是 CONCURRENCY 类:

```
Class CONCURRENCY{
```

```
protected:
    int proxy;
    char obj_name[8];
    REQUEST *request_queue;
    REQUEST current_request;
    ...
public:
    int put_args(ANY);
    ANY get_args(int);
    int init_port(int);//初始化服务员对象通信口
    int attach( ),attach(char *,int);
    int create( ),create(int),create(char *);
    int get_request( );
    int put_request( );
    int request( );
    int send_result(ANY);
    ANY result( );
    virtual void scheduler()=0;
};
```

其中attach方法有两种调用形式.一种为隐式调用attach( ),在执行远程方法请求时自动调用.另一种为显式调用attach(char *hostname, int port),由用户指明实际对象的主机号和端口号.显式调用适合于顾客/服务员模型的用户作业.

在进行远程方法请求时,首先要把参数转换成ANY类型,然后与方法名一起放入请求报文.转换过程由put_args利用ANY类中的构造函数自动完成.同样,在result返回结果时,利用操作符重载自动进行类型转换.由于目前使用的AT&T C++2.0版不支持模板(Template),对用户自定义的类不能进行自动转换.

3 运行支撑系统

用并发类库扩充的面向对象语言在实际应用中,一般都需要特殊的底层软件支持(运行支撑系统).对于不同的运行环境,这种底层支持的复杂性差别很大.单机上实现最容易;多处理机系统上较难;在松散耦合分布式环境中最难.底层支持可以建立在通用操作系统之上,也可建立在特殊的操作系统内核中,或者利用现有的某种软件平台.利用现有软件平台,简单易行,但由于它们绝大多数对并发面向对象编程方法缺少支持,系统性能受到限制.底层支撑建在特殊操作系统内核中,其实现效率高,但工作量大,可移植性差.本文运行支撑系统是在通用网络操作系统之上自行设计的,这样做,虽然工作量也比较大,但能与并发类库很好地配合,充分满足并发面向对象编程要求,并且便于移植.

本文运行支撑系统由本地前台管理员和各节点服务员组成,为用户作业中的并发对象分布执行提供支持.

· 前台管理员 由用户在本地(基地机)启动,负责接收本地用户的作业,并选择空闲机执行,管理远程I/O及处理异常事件.

· 节点服务员 驻留在所有机器上,后台运行.负责接收并执行用户基地机前台管理员发来的用户作业.管理在本地执行的用户作业及其生成的并发对象,与基地机前台管理员协同完成异常事件处理.

前台管理员在收到一个用户作业后,挑选一台空闲机发送此作业及作业信息,参见表1.远程空闲机后台服务员接收此作业后,记录作业信息,然后派生一进程执行此作业.用户作业执行过程中,首次创建并发对象时,向本地后台服务员查询本作业信息.每次创建并发对象时,根据host_i和host_n决定是否在本地创建,若需在远程机上创建,则给基地机前台管理员发出远程创建请求报文.基地机前台管理员在收到创建请求报文后,将发出请求的作业发送至另一机器上以创建相应并发对象.并发对象在创建后,首先把自身进程号等发给本地后台服务员,由后台服务员对并发对象进行统一管理.图3中,椭圆代表进程,长方形代表报文.

表1 作业信息格式

基地机主机名	基地机前台管理员通信端口号
空闲机顺序号 host_i	空闲机器数 host_n
……	

在运行支撑系统控制下,同一用户作业可同时在多台机器上执行,但创建的并发对象各不相同.创建全部对象只

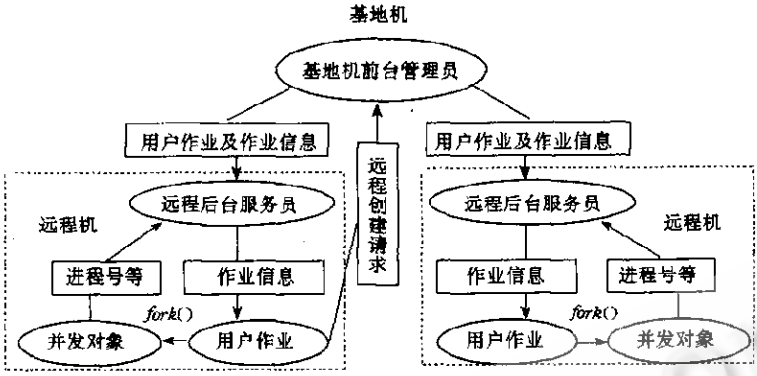

图3

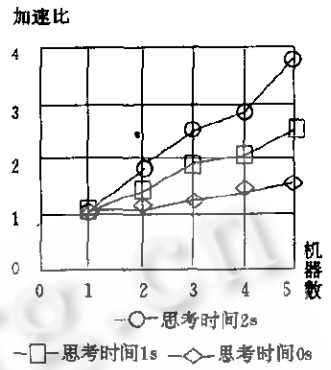

图4

需2 * min(objn,hostn) 1个报文.缺点是动态创建时,一个对象的实际创建数目可能不等于1,出现重复创建或不能创建的情况.解决办法是对 create 进行重载,在参数中规定逻辑节点号或实际机器名.形式:create(int i) $i>=0$,或 create(char * hostname).此方法不增加创建报文数,已在类库中实现.

运行支撑系统建立在 Sun OS 4.1.3上.异步通信使用网间域数据报型 socket 实现,远程 I/O 使用链接型 socket 实现.

在网络环境下,保证系统的稳定性比单机要困难得多,为把网络故障及个别机器崩溃的影响减至最小,我们主要采取以下措施.首先,在系统创建远程服务员时,基地机向客户机发出试探报文,同时等待客户机的回应报文,以检验客户机的可用性,将已崩溃的机器排除在外;其次,在远程服务员运行期间,基地机与客户机定期地交换承认报文,防止一个远程机崩溃后系统无限期等待.

4 结束语

本文研究环境为20台 Sun4工作站所组成的 Sunnet,操作系统为 Sun OS 4.1.3,基础语言为 AT&T C++2.0.研究工作是初步的,还有许多问题没有涉及.但是,通过用扩充后的 C++编写应用程序,在网中多台机器上分布并行,测试结果表明本文工作是切实可行的.系统在运行通信频繁的5个哲学家就餐问题时还表现出良好的加速性能,图4给出了加速比.

参考文献

1 Murat Karaorman, John Bruno. Introducing concurrency to a sequential language. Communication of ACM, 1993,36(9):103~116

2 Kai Töter, Carsten Hammer, Werner Struckmann. PARC++: a parallel C++. Software-Practice and Experience, 1995,25(6):623~636

3 Kafura, Mukherji M, Lavender G. ACT++, a class library for concurrent programming in C++ using actors. Journal of Object-Oriented Program, 1993,6(6):47~55,62

A Concurrent Class Library for C++

YANG Yan-zhong WANG Wei TIAN Lai-sheng

(*Department of Computer Science Jilin University Changchun* 130023)

Abstract In this paper, the authors discuss how to introduce concurrency into sequential object-oriented programming language. C++ as an example, the concurrency classes and corresponding tools are provided in the concurrency class library to support the distributed object-oriented programming. This paper shows the design and implementation of the concurrency class library and the supporting system. Last, the preliminary testing results are given.

Key words Concurrent object-oriented, concurrent object, run-time support system.