基于PVM的C++对象分布并行机制的初步研究*

李 毅[1], 周明天[2], 虞厥邦[1]

[1](电子科技大学 光电子技术系,四川 成都 610054);

[2](电子科技大学 计算机学院,四川 成都 610054)

E-mail: mtzhou@uestc.edu.cn

http://www.uestc.edu.cn

摘要: 对象具有内在的并行性.将面向对象程序设计与分布并行处理相结合,可产生既具有面向对象特征,又充分利用资源,还可缩短作业运行时间的对象分布并行系统.提出了一种基于PVM(parallel virtual machine)的C++对象的分布并行机制.该机制以协议和pvmlib作了向后兼容扩充的PVM系统为对象分布并行支撑;用预处理器将用户作业的并行类分离,并派遣到(PVM)系统中的目标机进行异地编译、加载执行;通过把并行类映射为PVM任务,请求对象消息映射为请求PVM任务消息来实现并行类对象的分布并行.实验结果表明,此对象分布并行机制(当问题规模达到一定程度时)可提高系统资源利用率和程序运行效率,并能简化PVM应用编程.

关键词: PVM(parallel virtual machine);面向对象;分布并行

中图法分类号: TP311 **文献标识码**: A

作为客观实体抽象的对象具有内在的并行性[1].将面向对象程序设计与分布并行处理结合所产生的对象分布并行系统,既具有面向对象特征,又可充分利用资源、缩短作业运行时间,还能大大降低分布并行程序的开发难度,因而是近年来分布式软件开发环境方面的研究热点.

文献[2~4]分别介绍了各自的并发C++的扩充技术及研究动态.但是,并发不等同于并行.没有分布并行环境的支撑,对象透明分布并行是难以实现的.从一般意义上讲,并发不能解决分布式系统环境下的网络资源利用问题.并且,由于并发机制的引入,系统开销增大,当作业的并发单元均为计算型时,运行效率低于顺序模型.文献[5,6]研究了共享对象的并行程序设计方法,文献[6]还介绍了分布共享对象数据一致性维护的Dual-Object技术.在此类并行面向对象语言中,对象共享基本是数据共享,且对象不是并行单位.文献[7]只给出了基于PVM的MPC++对象并行语言的并行类PVMPAR定义,并没有介绍编译器方面的工作、PVM支撑的研究工作以及对象分布和并行的机制.文献[8,9]介绍了对象可分布并行的C++编程系统.他们的方法是可行的,但也存在一些问题:首先,开发高效可靠的分布并行支撑系统的工作非常浩繁;其次,开发这种专用支撑系统的必要性值得斟酌.

PVM(parallel virtual machine)[10]是一个优秀的异构多机分布计算环境支持软件.Sun公司正与橡树岭国家实验室合作,以商品化该软件.目前称为"事实上的标准"的PVM被广泛用于基于一般局域网的分布并行计算.基于PVM进行软件研制和开发无论从学术角度还是商业角度出发都

* 收稿日期: 1999-08-03; 修改日期: 2000-04-13

基金项目: 国家自然科学基金资助项目(69871005);国家"九五"国防预研基金资助项目

作者简介: 李毅(1957—),男,山西交城人,博士生,副教授,主要研究领域为计算机操作系统,分布并行处理,分布对象技术;周明天(1939—),男,广西容县人,教授,博士生导师,主要研究领域为计算机网络,分布并行处理,分布对象技术,网络与信息安全;虞厥邦(1932—),男,广西桂林人,教授,博士生导师,主要研究领域为非线性动力学,CAD,混沌通信,神经网络.

是极有前途的.在研究它的可伸缩性的过程中,我们认为它以可作为性能优良的对象并行支撑.本文的主要研究工作是改进文献[8]的方法,在原PVM体系结构基础上,对PVM通信协议和lpvm.a编程接口作了向后兼容的扩充,为其增加了对象分布并行的支撑功能;同时,开发了专用的基于PVM的C++对象分布并行预处理器(任务).在普通C++编译器的支持下,首次在PVM系统上实现了真正的C++对象分布并行.功能扩充后的PVM完全保留原来的并行虚拟机管理功能,并可同时支持多个C++对象分布并行用户作业的运行.C++对象分布并行用户作业作为普通的PVM用户作业和分布并行对象、作为普通的PVM任务在PVM环境下运行.对象的分布、并行和通信由PVM管理,对用户完全透明.本文建议了这种在PVM系统上实现C++对象分布并行的新机制.

需要指出的是,本文介绍的对象分布并行机制不同于DOC(distributed object computing)的典型CORBA(common object requcst broker architecture)[11,12]的对象分布概念.这主要体现在3个方面:(1) CORBA的底层(infrastructure)是ORB(object request broker),而本文的机制的底层是PVM系统;(2) 在CORBA中,对象服务是系统事先确定的,对象接口(interface)先于客户程序存在于客户端,接口实现(implement)先于客户程序存在于服务端,而本文的分布对象由用户自己定义;(3) 在CORBA中,客户程序通过对象引用请求对象操作,其自身并不分布并行,而在本文的机制中,用户程序代码以并行类的方式分布并行.由此可见,本文的机制使用户具有较大的自主性和灵活性.

1 对象分布并行模型

1.1 改进的master-slave模型

只含有单向请求服务的master-slave模型在分布式系统中被广泛采用,如文献[2,7~10].本文采用一种改进的master-slave模型.改进主要体现在底层的实现机理上(参见第2.1节),对用户完全透明.由程序员的观点,系统的用户作业中主要有如下成员:

(1) master即主控任务,相当于文献[3]中的主动对象.由用户作业的main函数和所有相关对象(变量)、函数与类型定义集合组成.这里,相关是指直接引用,但排除并行类及其对象定义.

(2) slave即分布并行对象,是由新引入的关键字parallel class标识的类(并行类)所说明的对象.它的源码是并行类定义、相关函数和类型(基类、子类)的定义集合.逻辑上,它可以独立地分布并行;当无对象请求消息时,它处于睡眠等待状态;当请求消息到达时被唤醒,进行服务,最后发送结果消息给请求者.

(3) 被动对象即普通对象,只有在接收到外界的请求消息时,才被激活运行.它不能独立运行,串行和本地执行是其特征.

当用户作业主控任务启动时,所有并行类对象处于就绪状态,等待接收、处理请求消息.用户作业的主控任务向并行类对象发送请求消息,接收结果消息.主控任务在需要时向并行类对象发送析构命令消息,并行类对象执行相应的析构操作.

1.2 与分布并行对象通信

向分布并行对象发送的请求消息分为同步发送和异步发送.需要时阻塞接收两类.其语义参见文献[8].

(1) 同步消息

pp=parallel_obj.method_i (real_args);

pp=parallel_obj.member_var_j.

pp是指向结果消息缓冲的void*指针,结果消息缓冲区的结构为

asid	len	data

结果数据data的结构由请求方和服务方根据程序员的约定来处理.asid是异步消息标识,用于异步发送-需要时阻塞接收消息队列管理,以提高程序的并行程度.len是data的字节长度.对象方法请求中的实参real_args也按该方法处理.

(2) 异步发送-需要时阻塞接收消息

asid=asynsend(parallel_obj.method_i,real_args);

asid=asynsend(parallel_obj.member_var_j,NIL);

pp=blockrecv(asid).

异步发送-需要时阻塞接收消息是通过主控任务的lpvm.a的mroute,mhandle和blockrecv操作一个异步接收消息缓冲队列msg_list来实现的(见第3.3节的(4)中的②和④).

2 基于PVM的对象分布并行问题

2.1 对象-任务的转换

PVM的调度运行单位是任务.实现对象分布并行,需将分布并行对象(包括主控任务)转换为PVM任务,它才能接受PVM系统的管理和通信服务.在一般应用中,并行类的实例对象较少(很少多于1个),转换策略是将每个并行类作为一个PVM任务,通过并行类的分布并行来实现对象分布并行.逻辑上的分布并行对象的运行实际上映射为并行类对该对象请求消息的响应,即并行类任务在该对象实例上以消息为参数的一次执行.同一并行类的多个实例对象共享同一运行代码和PVM任务,但有各自的数据信息.并行类任务维护一个FIFO队列req_msg_list,用来存放对本类对象的请求消息(见第3.2节).该方法资源开销小,逻辑合乎实际,实现简单.当同一并行类的对象多于1个时,程序员又希望获得较高的并行度,他可以定义多个具有不同名字但有相同代码的并行类,令每个并行类只说明一个对象,便可解决并行度相对较低的问题.

2.2 请求对象消息-请求PVM任务消息的转换

为了便于处理对象请求消息,在预处理用户作业时,给每个并行类任务分配一个全局的任务号tid;为并行类的每个对象编一序号;为并行类的每个公有成员(公有方法或变量)编一个顺序号.其中0号用于析构函数,并将请求对象消息转换为请求PVM任务消息.该转换是如下映射的:

并行类对象.公有方法(实参表)——消息标识asid、PVM消息类型、并行类任务号tid、对象编号i、方法编号j、实参表.

并行类对象.公有成员——消息标识asid、PVM消息类型,并行类任务号tid、对象编号i、成员编号j.

PVM消息类型用于PVM的消息驱动机制,即指定目标pvmd用何协议函数处理该消息.

这种转换由预处理任务进行,转换后的消息由PVM收发.

2.3 并行类任务的派遣——远端编译和启动

PVM任务的tid是PVM系统生成(spawn)该任务时透明地为其分配的全局唯一的标识(用

于消息的寻址和任务管理),是运行时的信息.请求分布并行对象的消息是运行前的编译信息.为了能够正确地与分布并行对象进行通信,对象分布机构(预处理任务,见第3.1节)在生成并行类任务源代码时,用PVM系统功能先行获取该任务的tid,将其编码到源代码中;然后通过PVM消息机构把并行类任务源码派遣到目标机(tid中已指明),由远地pvmd对源码进行异地编译,加载运行.并行类代码启动后执行我们开发的startpvmtask(tid),通知pvmd本进程是任务号为tid的PVM任务,接受PVM的管理和通信服务.预处理时的消息转换(见第2.2节)使用该tid来指定请求对象消息的目标并行类任务.

3 基于PVM的C++对象分布并行的实现

3.1 预处理任务和用户作业主控任务

预处理任务有两个执行阶段.第1阶段,从用户作业源程序中分离并行类对象说明码,生成并派遣并行类任务源码;生成用户作业主控任务源码.第2阶段,创建子进程加载执行用户作业主控任务源码,启动主控任务.预处理任务如下:

```
main (argc,argv)
{   /* 第1预处理阶段 */
    执行 lpvm.a 函数 pvmbeatask(),使本进程成为 PVM 任务;
    由 argv 取用户作业源程序文件名 parobjsrc.cpp;
    分析、识别登记 parobjsrc.cpp 中的类、对象(并行类及其对象),并返回并行类个数 parclassn;
    while (parclassn 大于 0){
      向 pvmd 申请并行类任务号 tid;
      从 parobjsrc.cpp 中提取一并行类及对象说明码,并加入主控线索 main(见第3.2节),填写该并行类任
          务号 tid,把该并行类说明码转换为可运行的并行类任务码;
          用 lpvm.a 函数 pvm_send 将转换完毕的当前并行类任务码发送给 tid 所指定的主机 pvmd,消息
          类型为 DM_PAROBJ_START(并行类任务的启动见第3.3节的(1));
      把刚转换完毕的并行类说明码从 parobjsrc.cpp 中剔除,将 parobjsrc.cpp 的 main 函数中对当前并行类
          对象的析构请求改造为 PVM 消息发送,将当前并行类对象的直接请求(第1.2节的(1))改造为同
          步消息发送 synsend 调用,对 synsend,asynsend 和 blockrecv 调用的请求当前并行类对象消息进行
          请求 PVM 任务 tid 消息的转换(第2.2节),增加当前并行类各对象的初始化命令(PVM 消息);
      将尚未转换的并行类个数 parclassn 减1;
    }  /* 第2预处理阶段 */
    在 parobjsrc.cpp 启始时,添加 lpvm.a 的 pvmbeatask()函数调用命令;
    创建子进程;
    if(本进程是子进程)
        子进程编译(同 lpvm.a 连接)、加载执行 parobjsrc.cpp,启动主控任务;
    else
        父进程执行 lpvm.a 函数 pvmtaskexit(),退出 PVM 任务,并消亡;
}
```

改造后的用户作业主控任务是PVM任务(master),源码形式如下:

```
普通类和对象说明;
main(argc,argv)
{执行 lpvm.a 的函数 pvmbeatask(),使本进程成为 PVM 任务;
```

向每个并行类任务的各对象发初始化消息,PVM消息类型为DM_PAROBJ_REQ,请求tid并行类i对象的l初始化函数的消息发送为pvm_send(tid,DM_ PAROBJ_REQ(0,i,l,init_args));

消息收发和完成运算、处理的程序体;

tid并行类i对象作用域结束时向它发送析构函数(0号成员索引)请求:pvm_send(tid,DM PAROBJ REQ(0,i,0,nil));

}

3.2 并行类任务和并行对象

并行类任务作为独立的PVM任务(slave)运行,由并行类对象说明部分和控制线索main组成.并行对象说明部分包括相关基类、子对象类、方法定义及并行类对象数组parobj[n]说明.需说明的是,当并行类码被提取后,在生成并行类任务源码的同时,将新引入的关键字parallel_class改为class,以使普通C++编译器可以编译处理该源码.并行类对象说明部分如下:

```
与并行类相关的基类,子对象类说明;
class p_c_name { /* 并行类普通化:把parallel_class改为class */
    ...
    public:
      ...
      typei member_var_i;
      ...
      typej *method_j(args_list);
      ...
} parobj[n];   /* 共实例化n个本类对象 */
类成员函数定义;
```

并行类任务控制线索main由预处理器生成.其主要功能是由开关表根据对象号i和公有成员号j进行散转,完成本地pvmd转发来的主控任务的对象请求的处理,并送回结果消息.具体描述如下:

```
main()
{
 执行专门并发的PVM任务初始化函数startpvmtask(tid)(它不同于pvmbeatask),使本进程成为任务号为
      tid的PVM任务,tid为事先分配的任务号;
 安装信号处理函数parclasstidquit(见第3.3节的(4)中的③),其信号类型为SIG_PARCLASSTIDQUIT,以响
      应pvmd发送的任务结束信号(见第3.3节的(3)),即signal(SIG_PARCLASSTIDQUIT,par-
      classtidquit);
 while(true){
  if(请求对象消息队列req_msg_list空){
    安装信号处理函数wake_parclasstid,信号类型为SIG_WAKE,以响应pvmd发送的唤醒号,即
        signal(SIG_WAKE,wake_parclasstid);
    睡眠,等待被pvmd的SIG_WAKE信号唤醒;
  }
  从req_msg_list队列中取主控任务发来的请求$i$并行对象$j$公有成员消息
    TM_PAROBJ_REQ(asid,$i$,$j$,real_args);
  switch(根据公有成员号$j$进行散转){
    0:$i$并行对象析构,执行parobj[$i$].~$p$_c_name();
```

向 pvmd 发TM_PAROBJ_DESTR 消息报告一个对象已析构,即 pvm_send(pvmd,
TM_PAROBJ_DESTR(tid));
break;
...
l:请求 *i* 并行对象的 *l* 方法,并返回指向结果数据的 void 型指针 *p*,
即 *p*=(void *)parobj[*i*].method_*l*(real_args);
计算结果数据长度 len,即 len=sizeof(type*l*);
break;
...
k:请求 *i* 对象的 *k* 成员变量,该成员变量的(void *)地址为 *p*=(void *)&parobj[*i*].member_var_*k*;
取该成员变量的字节长度 len,即 len=sizeof(type*k*);
break;
...
}
将异步消息标识 asid、结果消息长度 len 和 *p* 所指的字节串(长度为 len)填入 pp 结果缓冲区,形成类型为 TM_PAROBJ_RESULT 的消息,其目标地址为从 req.msg_list 取出消息的源地址(主控任务 tid),用 pvm_send 将该消息直接发送到主控任务;
调用lpvm.a 底层函数 mroute(0,0,0,0),该函数检查本任务同本地 pvmd 的 TCP 连接口是否有消息到达,如果有,则调用安装在 mhandle 中的 TM_PAROBJ_REQ 类型(对于主控任务是 TM_PAROBJ_RESULT)消息处理函数(见第3.3节的(4)中的①和②),将消息逐一放入 req_msg_list 队列;
}
}

如果 req_msg_list 为空,则并行类任务睡眠.当请求任务对象的消息到达目标主机时,pvmd 将消息通过 TCP(transmission control protocol)连接发给该任务,同时向该任务发 SIG_WAKE 信号(见第3.3节的(2)).并行类任务(进程)被唤醒后,执行下列信号处理函数(用 mroute 函数查收消息):

```
wake_parclasstid(){mroute(0,0,0,0);}
```

3.3 PVM 的对象分布并行支撑

(1) 并行类任务启动

由 PVM 的消息驱动机制,当 pvmd 收到预处理任务向本 host 发送的 DM_PAROBJ_START 消息(见第3.1节)后,执行下面已安装到 ddpro.c 协议中的函数:

```
dm_parobj_start(要启动的并行类任务号 tid,并行类对象个数 objn,并行类任务源码长度 srclen,指向并行类
        源码起址的字节指针 srccode)
{   产生子进程并取子进程 pid;
    if(本进程是子进程)
        子进程编译(同 lpvm.a 连接)、加载执行并行类任务码;
    else
        父进程维护并行类任务-进程映射表 objtid_pid_tab,将 tid,pid 和 objn 填入一个空表项内;
}
```

(2) 对分布并行对象请求、应答的协议操作

主控任务发来请求并行类任务 tid 的 *i* 对象 *j* 公有成员(tid,DM_PAROBJ_REQ(asid,*i*,*j*,

real_args))消息,pvmd 执行以下已安装在 ddpro.c 的协议操作:

dm_parobj_req(异步消息号 asid,对象号 i,公有成员号 j,实参表 real_args)
{ 从 objtid_pid_tab 表中取 tid 所对应的 pid;
将消息类型改为 TM_PAROBJ_REQ 后,送 pvmd 的 task[tid]的 t_txq 队列,由 pvmd 的 work 循环中的 locloutput 透明地通过 TCP 口发到并行类任务 tid;
向并行类任务 tid 进程发出唤醒信号 SIG_WAKE,使之醒来后执行 mroute,查收对象请求消息(见第3.2节的 wake_parclasstid 信号处理函数),即 kill(pid,SIG_WAKE);
}

(3) 并行对象析构

并行对象 init 和析构请求的处理同本节的(2).0公有成员索引用于析构,并行对象执行完析构函数以后,给本地 pvmd 发 TM_PAROBJ_DESTR 消息(见第3.2节中的 main 控制线索),pvmd 收到消息后执行如下已安装到 tdpro.c 协议中的函数:

tm_parobj_destr(已执行对象析构操作的并行类任务号 tid)
{ 将本主机 pvmd 维护的并行类任务-进程映射表中 tid 项的并行类对象数减1,
即:objtid_pid_tab[tid].objn;
if (tid 并行类任务的对象已全部析构,即 objtid_pid_tab[tid].objn<=0){
向并行类任务 tid 进程发终止任务信号,使其在信号处理函数 parclasstidquit 中执行 pvmtaskexit(见第3.2节中的 main 控制线索和本节的(4)中的③),即 kill(objtid_pid_tab[tid].pid,SIG_PARCLASSTIDQUIT);
删除在并行类任务-进程映射表中的 tid 并行类任务表项:objtid_pid_tab[tid];
}
}

(4) lpvm.a 的扩充

在 ddpro.c 和 tdpro.c 中增加支撑协议函数后,由 PVM 的消息驱动机制,无论是主控任务或是并行类任务(不论它们是在不同主机上并行,还是在相同主机上并发),pvmd 均可正确、协调地处理或转发所收到的有关对象分布并行的 DM_类和 TM_类消息,如本节的(1)~(3)所描述.作为 PVM 任务独立运行的主控任务和并行类任务与本地 pvmd 建立连接后,由 pvmd 之间通过 tid 寻址的可靠 UDP(user datagram protocol)将它们联系起来.PVM 任务同本地后台进程 pvmd 的接口是 lpvm.a 库.为了使主控任务和并行类任务之间能正确通信和同步,我们必须将一些有关的分布并行对象任务同本地 pvmd 的专用接口函数一起添加到 lpvm.a 中.

① 给 lpvm.a 增加 recvmethreq(msg)函数,并行类任务的 startpvmtask 安装它到 mhandle.该函数处理 pvmd 发给并行类任务的 TM_PAROBJ_REQ 消息,将消息送入并行类任务的 req_msg_list队列.

② 在 lpvm.a 中增加 recvresult(msg)函数,将其安装到主控任务的 mhandle.该函数处理由并行对象发来的(主控任务 tid,TM_PAROBJ_RESULT(result))结果消息(其中的 asid 不为0),将消息送入 msg_list[asid]中,由 blockrecv 取出(见下文的④).

③ 在 lpvm.a 中增加 parclasstidquit 和 wake_parclasstid 信号处理函数.

parclasstidquit(){执行 lpvm.a 函数 pvmtaskexit(),终止本任务(进程);}

wake_parclasstid 信号处理函数见第3.2节,在此不重复.

④ 消息改造后的消息收发函数:同步消息发送-接收函数 synsend、异步发送-阻塞接收函数

asynsend 和 blockrecv 也被添加到 lpvm.a 中.

```
void *synsend(目标并行类任务号 tid,对象 i,公有成员 j,实参表 real_args)
/* 当请求成员 j 变量时,real_args 为 NIL,下同 */
{ 把参数 i,j,real_args 打包,生成 DM_PAROBJ REQ 类型的 PVM 消息;
  以本任务号为源址,用 lpvm.a 同步发送阻塞接收函数把消息发向目标 tid,其中 asid 为0,即
        sndrcv(tid,DM_PAROBJ_REQ(0,i,j,real_args));
  取 sndrcv 返回的结果指针 p;
  return p;
}
int asynsend(目标并行类任务号 tid,对象 i,公有成员 j,实参表 real_args)
{ 申请异步消息号 asid; /* asid≠0用于同步发送和无返回发送(init,析构) */
  在本主控任务中的 msg_list 表用 asid 登记该消息发送;
  把参数 asid,i,j,real_args 打包,生成类型为 DM_PAROBJ_REQ 的 PVM 消息,以本任务号为源地址,tid
  为目标地址,用 lpvm.a 函数发送该消息,即 pvm_send(tid,DM_PAROBJ_REQ(asid,i,j,real_args));
  return asid;
}
```

由 synsend 和 asynsend 函数发送的请求对象消息均由目标主机的 pvmd 根据协议转发给目标并行类任务 tid(见第3.3节的(2)以及第3.2节).

```
void *blockrecv(阻塞接收的异步消息号 asid)
{
 while (如 asid 标识的本主控任务 msg_list 表项空){
  定时睡眠,等待若干时间;
  执行mroute(0,0,0,0),查收 TM_PAROBJ_RESULT 消息,若有,则放入 msg_list 相应的表项中(见
      第3.3节的(4)中的②和第3.2节的 mroute 说明);
 }
 从 asid 标识的 msg_list 表项中取结果消息指针送 pp,并将其返回;
}
```

3.4 测试实例

我们选用文献[8]的测试例题(求 $\sum_{i=1}^{n} i*i*i*i$),在1台586/200(主 host)和3台486/66(从 host)构成的 PVM 环境下对对象分布并行系统进行了编程测试.系统在理想的轻载条件下的测试结果见表1和表2.

Table 1 The time table of test example compilation and parallel-class separation (preprocessing)

表1 测试例题的编译、并行类分离(预处理)时间表

	Compilation time of master task/parallel-class task (s)①	Packet transfer time (ms)②	Time of preprocessing three parallel-class tasks (s)③	Time of preprocessing four parallel-class tasks (s)④
486 slave host⑤	1.6	9.1		
586 master host⑥	0.6	0.5	4.9	6.5

①主控任务/并行类任务编译时间(秒),②报文传送时间(毫秒),③3个并行类任务预处理时间(秒),
④4个并行类任务预处理时间(秒),⑤486从机,⑥586主机.

从表1可以看出,在网络轻载条件下,报文传送时间相对于运行时间很小,可以忽略;预处理时间(不含主任务编译时间)随着并行类个数的增加而增加.表2的结果表明,并不是并行对象越多,加速比就越大.当 $n \geqslant 50000000$ 时,4个对象并行的运行时间明显缩短,这说明只有当问题规模达到一

定程度,对象分布并行获得的加速增益完全抵消主控任务(串行)预处理全部并行类的开销(与并行类个数及其代码长度有关)后还有盈余时,系统的并行优势才能显示出来.

Table 2 The running time contrast table (s)

表2 运行时间对比表 (s)

n	An object (single host) serial running①	Three objects parallel running②	Four objects parallel running③
1 000 000	5.2	8.1	9.1
5 000 000	17.3	12.7	12.9
10 000 000	33.2	18.3	19.7
50 000 000	156.1	59.6	46.6
100 000 000	312.5	110.2	86.8

①1个对象(单机)串行,②3个对象并行,③4个对象并行.

4 结束语

本文所做工作的主要成果有:(1) 采用预处理技术将支持C++ parallel class并行类和master_slave模型的源程序映射为可在PVM环境下分布并行的对象任务源码集合;(2) 利用消息驱动机制成功地分布(派遣)、异地编译运行了分布并行对象,并以客户-服务器模型和PVM消息实现了主控任务对分布对象公有成员的访问;(3) 在PVM内核增加了分布并行对象协议和相应的lpvm.a接口,开发了分布并行对象的支撑.

工作的主要贡献在于:(1) 基于PVM支撑,实现了一种新的对象分布并行机制,扩展了PVM系统的功能;(2) 提高了C++程序对资源的利用率和运行效率;(3) 简化了PVM应用编程,利用请求对象消息和对象分布并行,对程序员完全屏蔽了PVM的消息收发和任务管理.

在课题的设计和实现中,为验证基于PVM的C++对象分布并行的主导思想,对原问题进行了简化.为了使系统功能进一步完善,现在进行的研究工作有:

(1) 对象分布并行系统中类继承和函数重载的一般处理技术(包括交叉继承问题);

(2) 并行类对象的动态生成问题;

(3) 并行类指针和并行对象引用问题.

References:

[1] Nelson, M. L. Concurrent & object_oriented programming. ACM SIGPLAN Notices, 1991,26(10):63~72.

[2] Chen, Jia-jun, Zhao, Jian-hua, Zheng, Guo-liang. An approach to concurrent extension of C++. Journal of Software, 1998,9(8):586~591 (in Chinese).

[3] Yang, Da-jun, Zhang, Ming, Lu, Jian. The study of concurrent object-oriented programming languages. Computer Research & Development, 1998,35(9):769~775 (in Chinese).

[4] Wen, Dong-chan, Wang, Ding-xing, Zhang Ning. CCPP: the design and implementation of a concurrent C++ language. Journal of Software, 1997,8(6):401~408 (in Chinese).

[5] Ji, Xue-rong, Wen, Dong-chan, Wang, Ding-xing. Design and implementation of a shared-object-based SOC++ parallel language. Computer Research & Development, 1997,34(supplement):84~88 (in Chinese).

[6] Yuan, Wei, Sun, Yong-qiang. Dual-Object approach to object oriented parallel programming. Journal of Software, 1998, 9(1):47~52 (in Chinese).

[7] Wen, Dong-chan, Wang, Ding-xing. The implementation of parallel C++ languages for workstation clusters. Chinese Journal of Computers, 1997,20(1):9~17 (in Chinese).

[8] Tian, Lai-sheng, Dong, Zhe, Su, Xi-yong. DOPS-- distributed objec: oriented programming system. Mini-Micro Systems, 1997,18(7):29~35 (in Chinese).

[9] Yang, Yan-zhong, Wang, Wei, Tian, Lai-sheng. A concurrent class library for C++. Journal of Software, 1998,9(6): 401~404 (in Chinese).

[10] Geist, A., Beguelin, A., Dongarra, J., *et al*. PVM: Parallel Virtual Machine——a Users' Guide and Tutorial for Networked Parallel Computing. Cambridge, MA: MIT Press, 1994.

[11] Orfali, R., Harkey, D., Edwards, J. Instant CORBA. New York: John Wiley & Sons, Inc., 1977. 53~72.

[12] Stallings, W. Operating Systems: Internals and Design Principles. 3rd ed. Upper Saddle River, NJ: Prentice Hall, Inc., 1998. 715~717.

附中文参考文献:

[2] 陈家骏,赵建华,郑国梁.C++的一种并发扩充方案.软件学报,1998,9(8):586~591.

[3] 杨大军,张鸣,吕建.并发面向对象程序设计语言研究与进展.计算机研究与发展,1998,35(9):769~775.

[4] 温冬蝉,王鼎兴,张宁.CCPP:一个并发C++语言的设计与实现.软件学报,1997,8(6):401~408.

[5] 计学荣,温冬蝉,王鼎兴.基于共享对象的SOC++并行语言的设计与实现.计算机研究与发展,1997,34(增刊):84~88.

[6] 袁伟,孙水强.Dual-Object:面向对象的并行程序设计.软件学报,1998,9(1):47~52.

[7] 温冬蝉,王鼎兴.基于机群系统的C++语言并行化实现.计算机学报,1997,20(1):9~17.

[8] 田赣声,董哲,苏希勇.DOPS——分布式面向对象编程系统.小型微型计算机系统,1997,18(7):29~35.

[9] 杨延中,王为,田赣声.具有并发类库的C++.软件学报,1998,9(6):401~404.

Preliminary Research on C++ Object Distributed-Parallel Mechanism Based on PVM*

LI Yi[1], ZHOU Ming-tian[2], YU Jue-bang[1]

[1](*Department of Opto-Electronic Technology, University of Electronic Science and Technology of China, Chengdu 610054, China*); [2](*College of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 610054, China*)

E-mail: mtzhou@uestc.edu.cn

http://www.uestc.edu.cn

Abstract: The object possesses inherent parallelity. Combination of object-oriented programming with distributed-parallel processing will bring about object-oriented distributed parallel system, not only having object-oriented property but also making better use of system resources and shortening user's computing time as well. In this paper, a novel C++ object distributed-parallel mechanism is proposed based on PVM (parallel virtual machine). The object distributed-parallel mechanism is supported by the PVM system whose protocol and pvmlib have been made backward compatible extension. It uses preprocessor to separate the parallel classes from user's job program and dispatches them to host computers in PVM to compile and run there. Through mapping parallel-class to PVM task, request object message to request PVM task message, the mechanism implements object distributed-parallelism of the parallel-class. The results of the experiment show that (when the size of question is big enough) the mechanism may make better use of the system resource, run user program efficiently, and simplify the PVM application programming.

Key words: PVM (parallel virtual machine); object-oriented; distributed-parallel

* Received August 3, 1999; accepted April 13, 2000

Supported by the National Natural Science Foundation of China under Grant No. 69871005; the Defence Pre-Research Project of the National 'Ninth Five-Year Plan' of China