# 计算科学与工程中的并行编程技术

# Parallel Programming Technology in Computational Science and Engineering

都志辉

清华大学计算机系

Email：duzh@tsinghua.edu.cn

Phone: 62782530

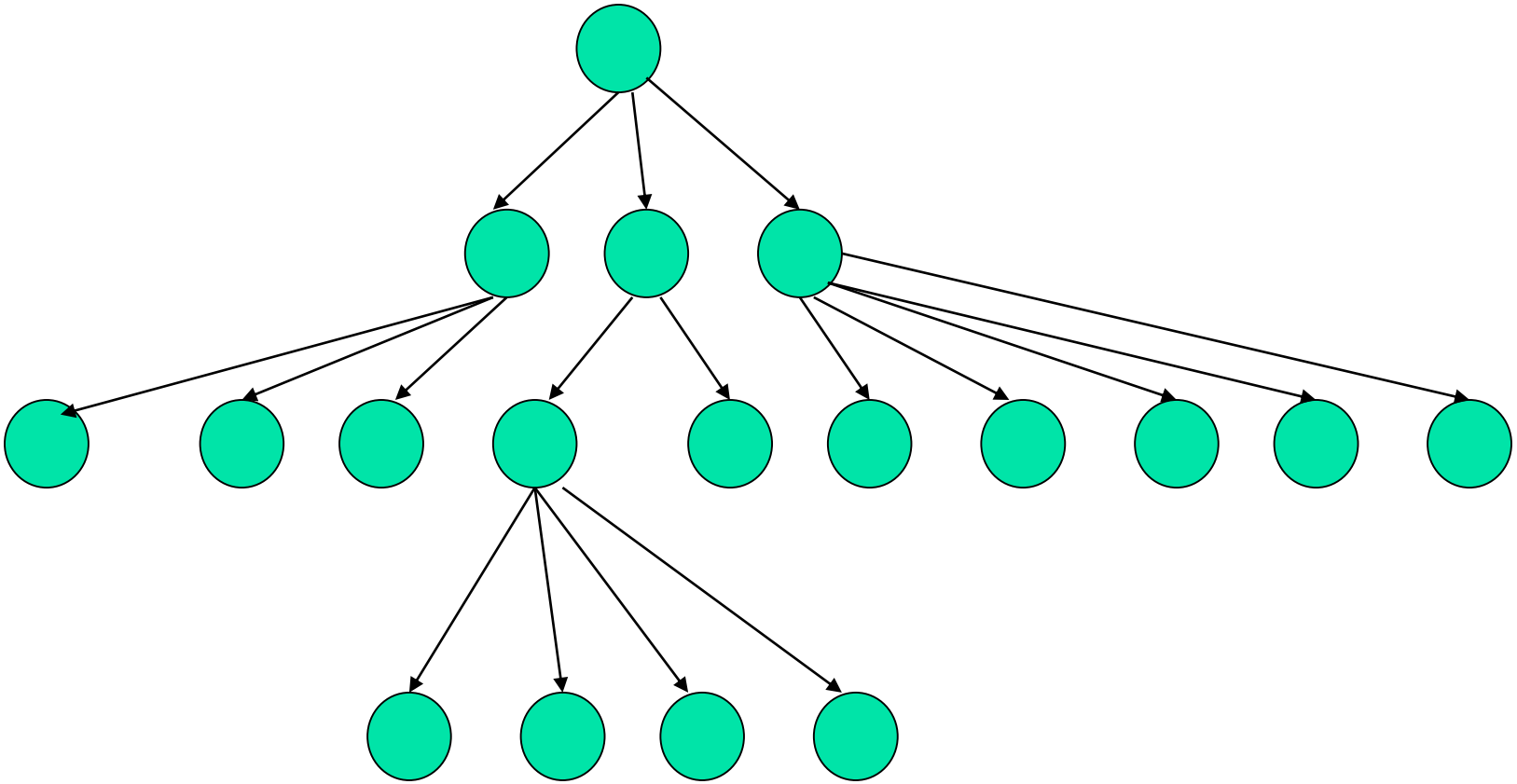
http://hpclab.cs.tsinghua.edu.cn/~duzh

# MPI－2的动态进程管理与单边通信

# MPI-1的不足

- 不支持进程个数的动态改变
- 不支持单边通信模式
- 不支持并行文件操作

# 应用需求

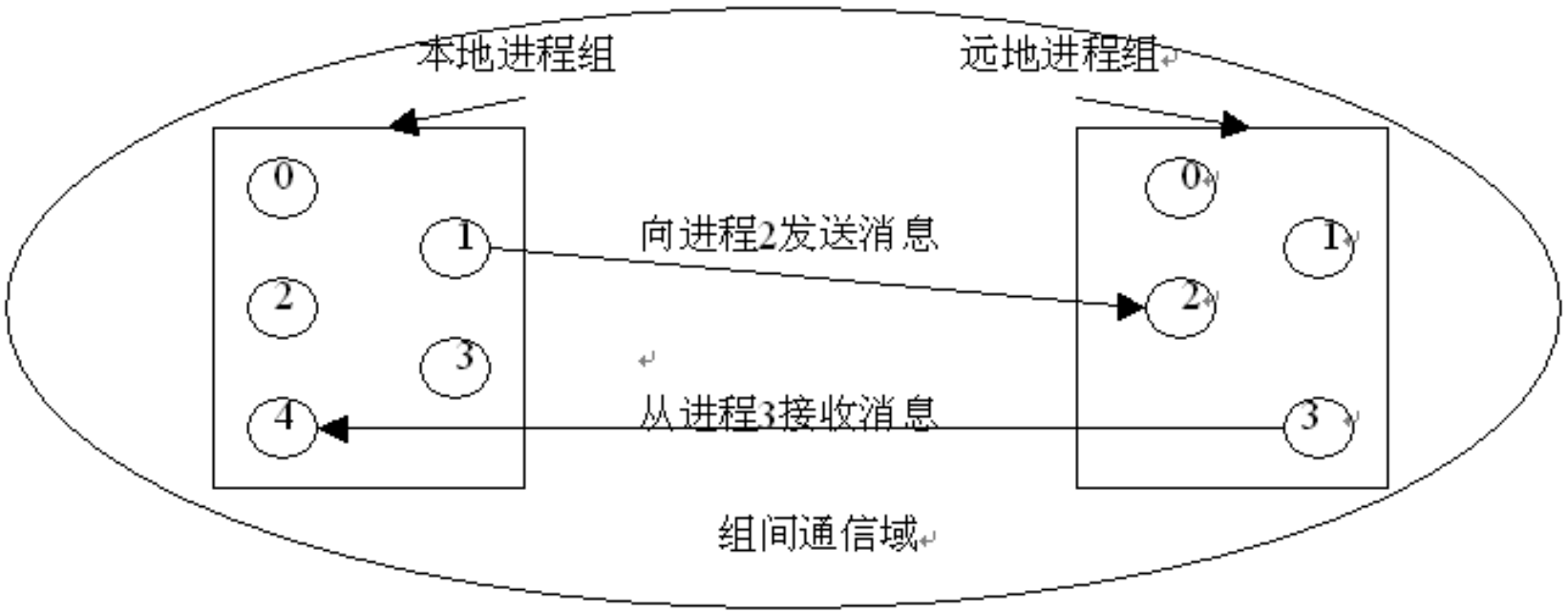


- 动态任务树
- 精确成型应用实例

# MPI-2的解决方案

- 对通信域进行扩展
  - 组内通信域
  - 组间通信域
- 具体实现方式
  - 动态派生进程（有父子关系）
  - 独立进程间通信（C/S关系）
  - Socket通信（转换socket通信）

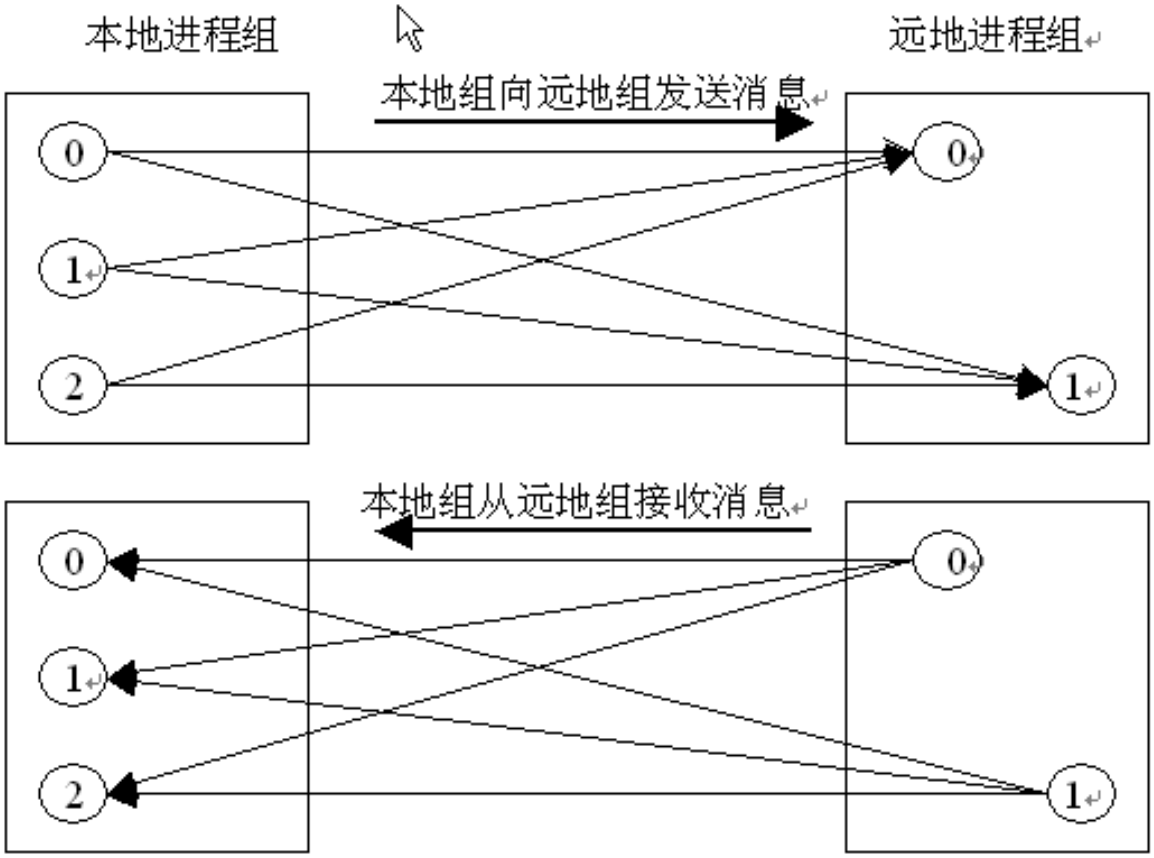
# 组间通信域的点到点通信



# 例子

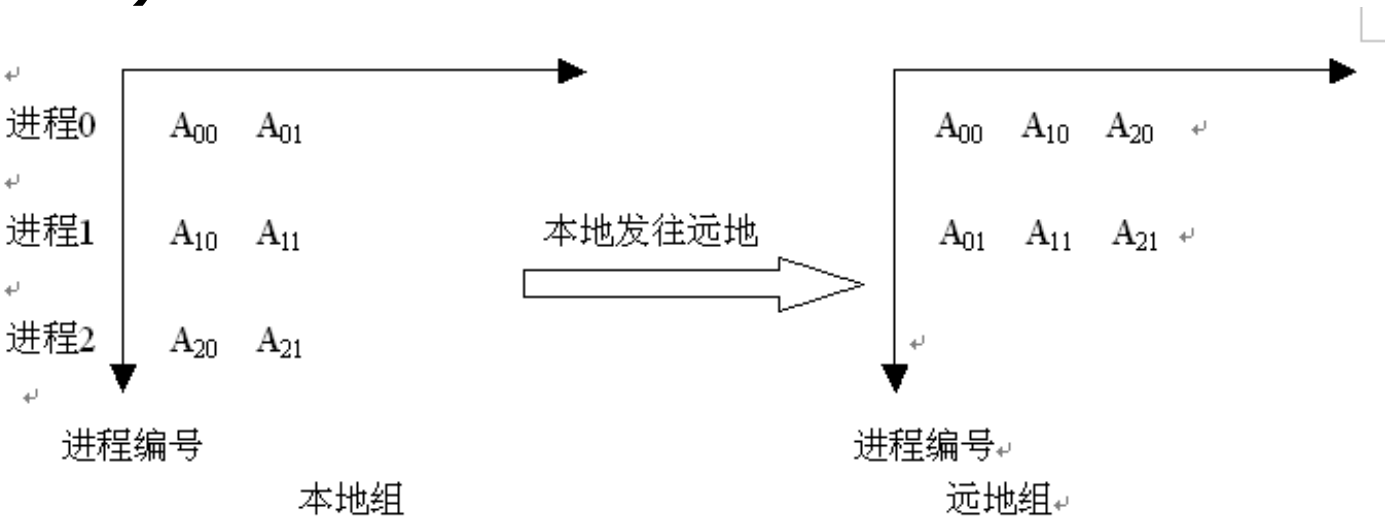
- MPI_SEND(buf,count,datatype,dest,tag,intercomm)
- 与组内通信的不同：1 dest的含义；2 intercomm
- MPI_RECV(buf,count,datatype,source,tag,
  intercomm,status)
- 与组内通信的不同：1 source的含义；2 intercomm

# 组间通信域的组通信

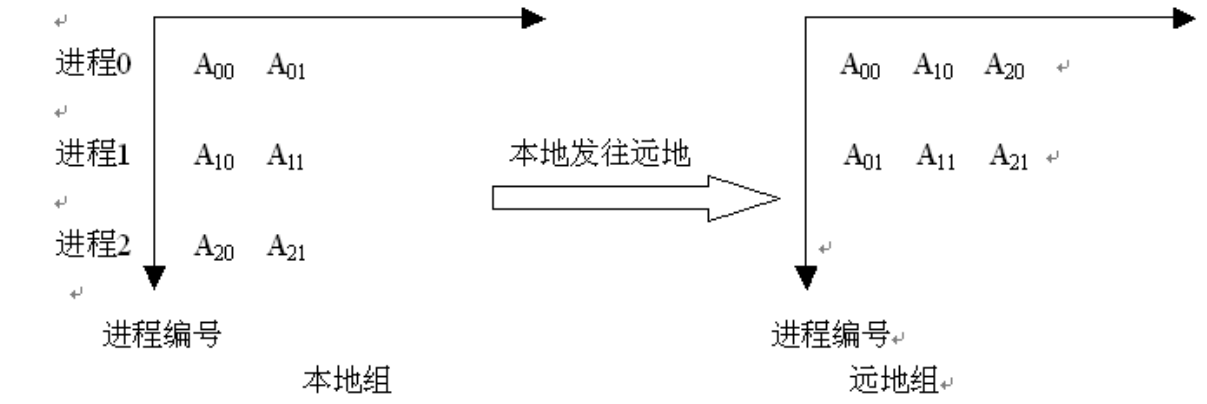


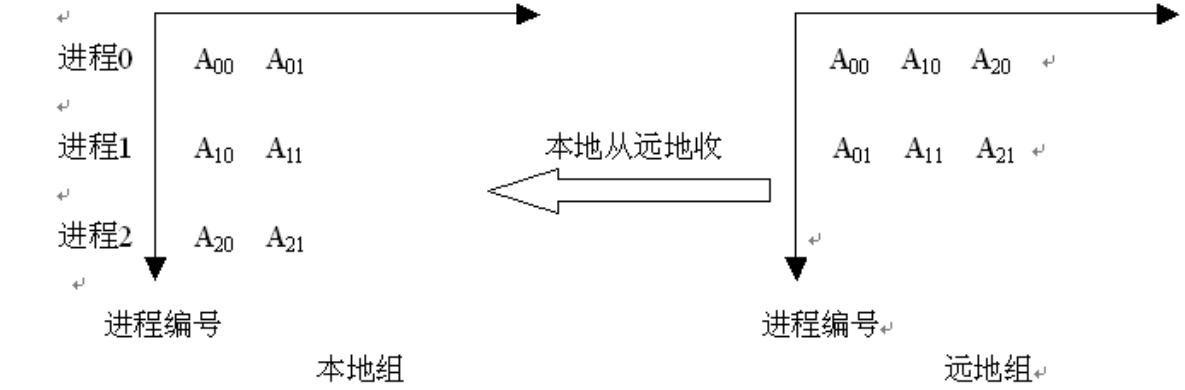
# 例子

- MPI_ALLTOALL(sendbuf,sendcount,sendtype,recvbuf,recvcount,recvtype,intercomm)



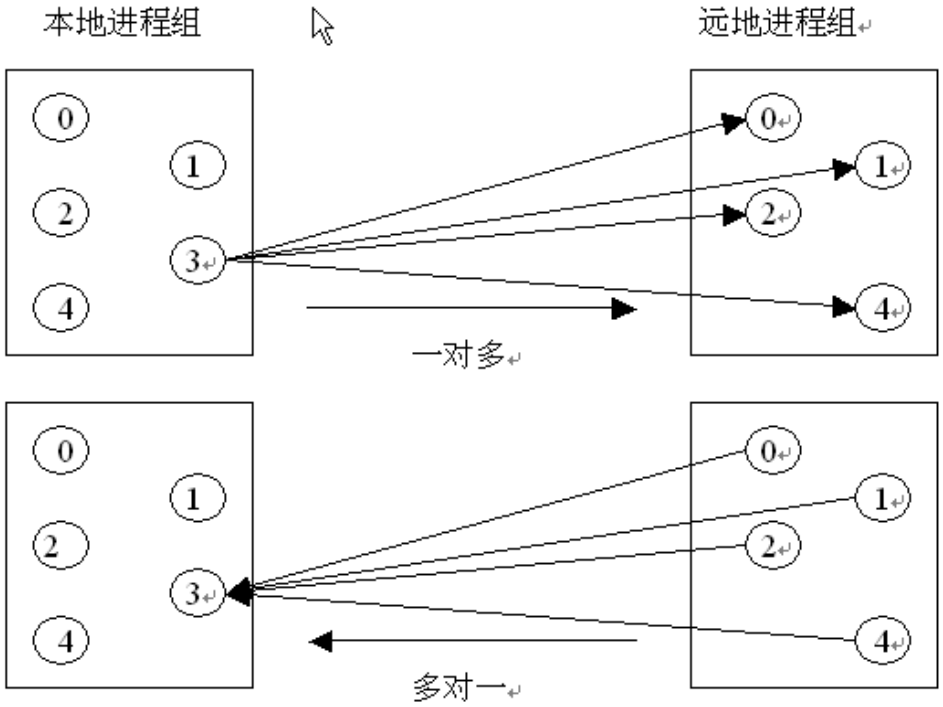
# 图示


进程0
进程1
进程2
A00 A01
A10 A11
A20 A21
本地发往远地
A00 A10 A20
A01 A11 A21
进程编号
本地组
进程编号
远地组


进程0
进程1
进程2
A00 A01
A10 A11
A20 A21
本地从远地收
A00 A10 A20
A01 A11 A21
进程编号
本地组
进程编号
远地组

# 组间通信域的一对多和多对一

- 本地一个发，对方所有的收
- 本地一个收，对方所有的发



# 例子

- MPI_BCAST(buf,count,datatype,root, intercomm)
  - ROOT所在的Local部分
    - MPI_BCAST(buf,count,datatype,MPI_ROOT,intercomm)
    - MPI_BCAST(buf,count,datatype,MPI_PROC_NULL,intercomm)
  - 与ROOT对应的Remote部分
    - MPI_BCAST(buf,count,datatype,,intercomm)

也即，不参与通信的local如何？

# 例子

- MPI_BCAST(buf,count,datatype,MPI_ROOT,intercomm)
- 与组内通信的不同：1 dest的含义；2 intercomm
- MPI_RECV(buf,count,datatype,source,tag,
  intercomm,status)
- 与组内通信的不同：1 source的含义；2 intercomm

# 主进程组中动态进程的创建

- MPI_COMM_SPAWN(cmd,argv,maxprocs,

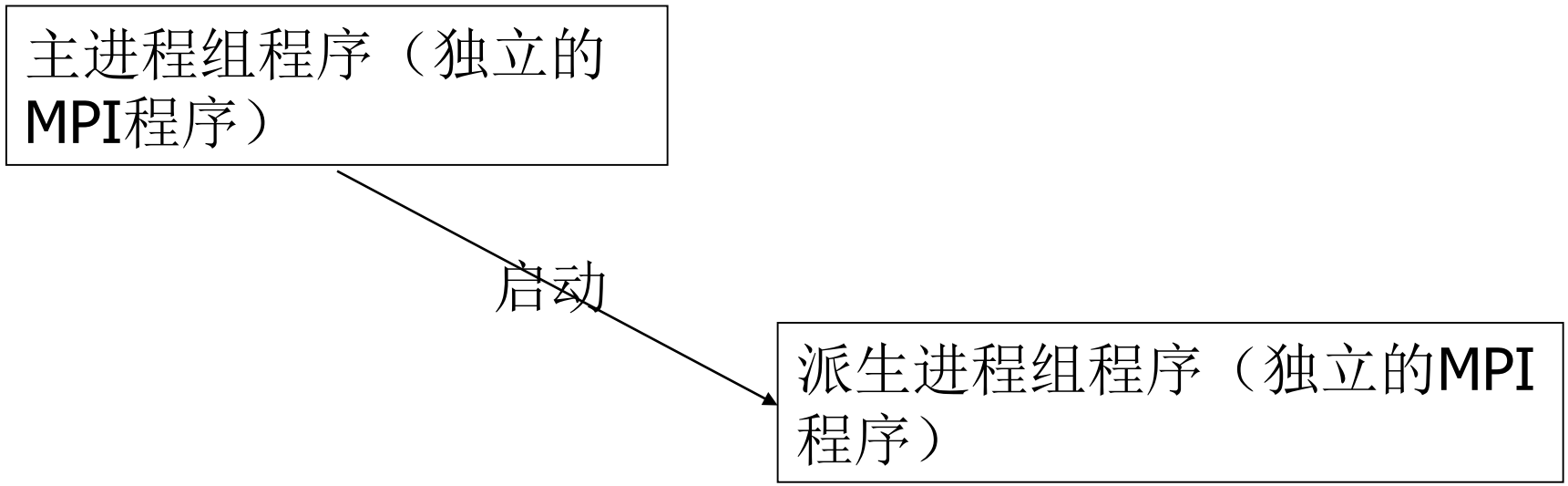
  info,root,comm,intercomm,aryerrcodes)

最多派生maxprocs个进程，root是参数检查的进程

# 派生进程组得到组间通信域

- MPI_COMM_GET_PARENT(parent)
- 在派生进程组中调用
- 在初始化之后
- 三次同步
  - 父进程同步
  - 子进程之间同步
  - 父子进程之间同步

# 源程序



# 创建多组子进程

- MPI_COMM_SPAWN_MULTIPLE(count,cmdary,argvary,maxprocsary,infoary,root,comm,intercomm,errcodeary)
- 各组子进程是不同的

# 独立进程间通信

- 服务端
- MPI_OPEN_PORT(info,port_name)
- MPI_COMM_ACCEPT(port_name,info,root,comm,intercomm)
- 在打开的特定端口上等待连接

# 独立进程间通信（续）

- 客户端
- MPI_COMM_CONNECT(port_name,info,
  root,comm,intercomm)
- 请求与指定端口连接
- MPI_COMM_DISCONNECT(comm)

# 问题

- 客户端每次连接的端口可能是不同的

# 方案

- 服务端
  - 建立特定名字和端口字符串的联系
  - MPI_PUBLISH_NAME(servicename,info,portname)
- 客户端
  - 根据名字得到特定的端口字符串
  - MPI_LOOKUP_NAME(servicename,info,portname)

# 基于socket的通信

- MPI_COMM_JOIN(fd,intercomm)
- 将socket通信转化为MPI通信

# 什么是远程存储访问（单边通信）？

Remote Memory Access/One-Sided Communication

- 打电话
- 留字条
- 阅读EMAIL

# 通信窗口

- 开辟自己内存的一段空间作为对外联系的窗口，任何进程都可以通过对窗口的访问来达到数据通信的目的

# 窗口操作的分类

- 窗口读 MPI_GET
- 窗口写 MPI_PUT
- 窗口运算 MPI_ACCUMULATE

# 窗口的创建

- MPI_WIN_CREATE(base, size, disp_unit, info, comm ,win)
  - Base：起始地址
  - Size：大小，字节为单位
  - Disp_unit:单位偏移，字节为单位

# 注意

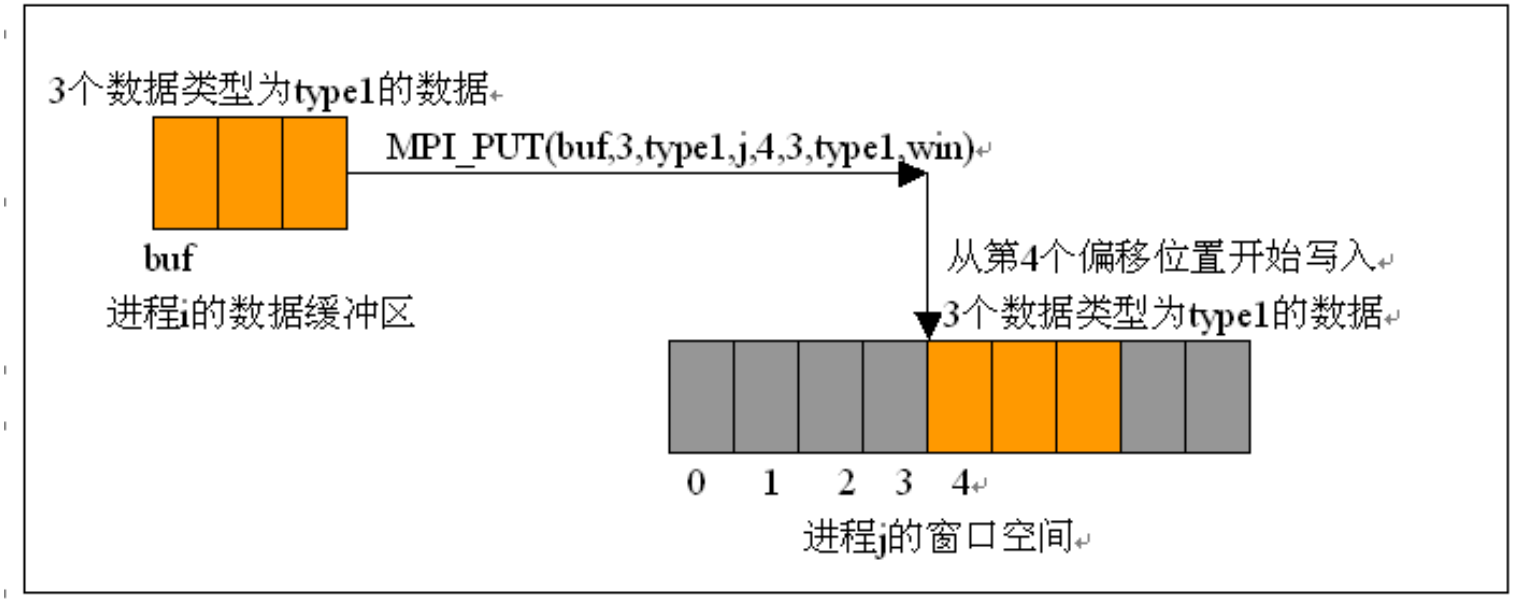
- 窗口创建操作是组调用，所有的进程必须都执行
- 得到的窗口是组窗口，与给定的通信域密切相关
- 用同一窗口对象可以访问所有其它进程的窗口

# 窗口的释放

- MPI_WIN_FREE(win)
- 当所有窗口都不再使用时释放
- 是组调用

# 远程更新（向窗口写）

- MPI_PUT(origin_addr, origin_count, origin_datatype, target_rank, target_disp, target_count, target_datatype, win)

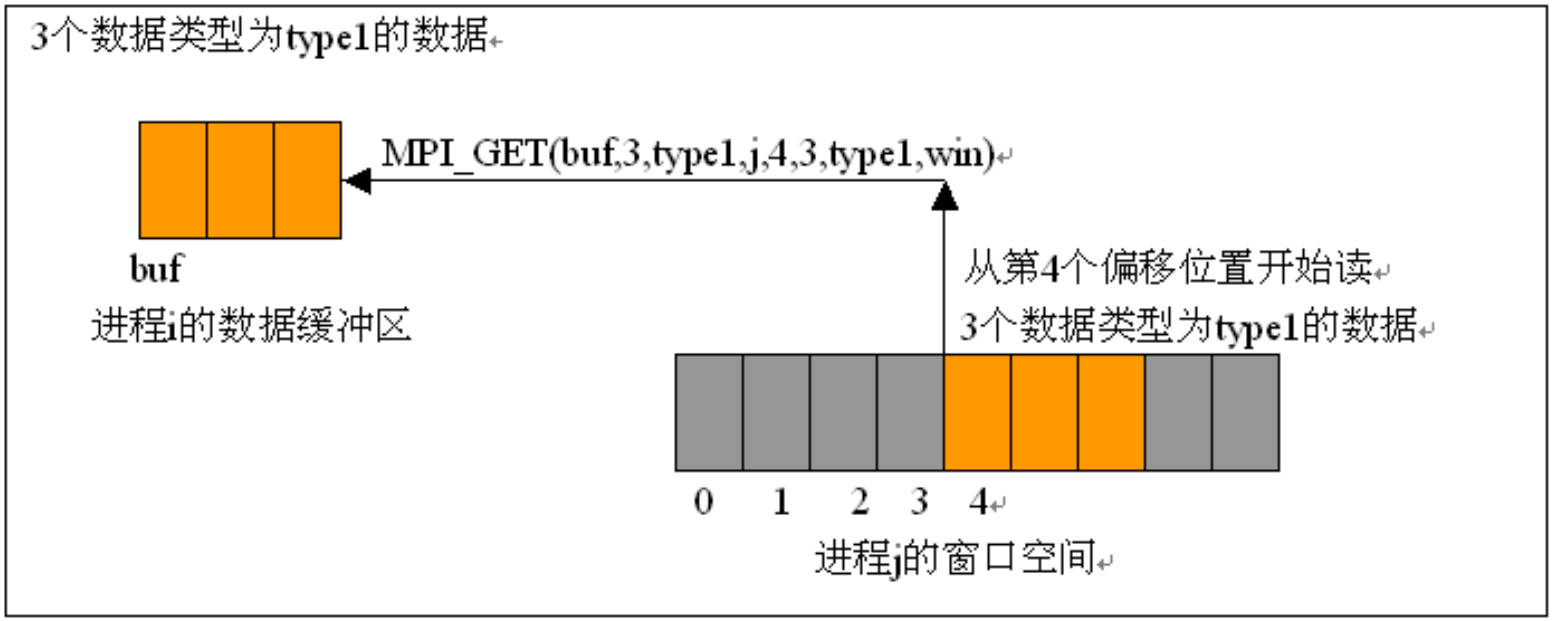


# 注意

- 不同进程的窗口是通过进程标识来区别的
- 对不同进程窗口的操作使用的是同一窗口句柄
- 偏移是以创建窗口时指定的偏移单位大小计算的

# 从远程得到数据（从窗口读）

- MPI_GET(origin_addr, origin_count, origin_datatype, target_rank, target_disp, target_count, target_datatype, win)

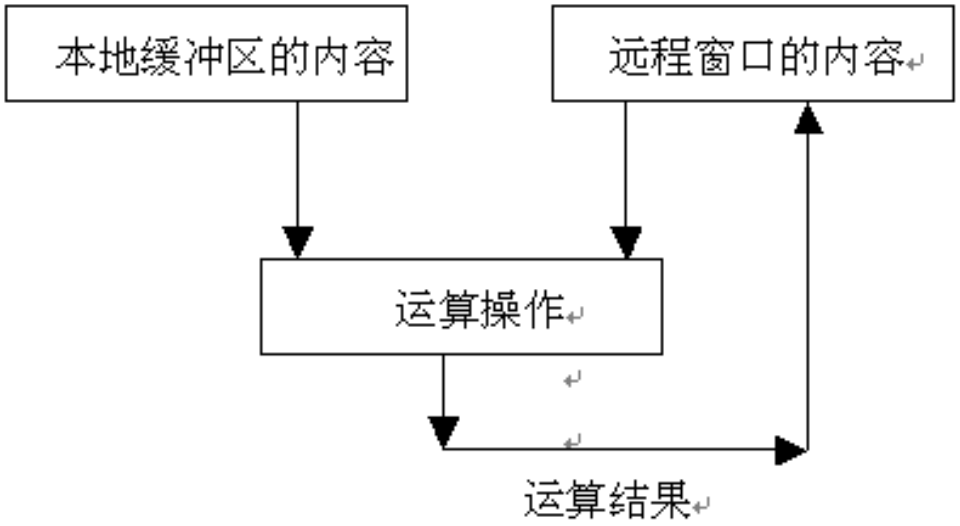


# 注意

- 写窗口的逆操作
- 不同进程的窗口是通过进程标识来区别的
- 对不同进程窗口的操作使用的是同一窗口句柄
- 偏移是以创建窗口时指定的偏移单位大小来计算的

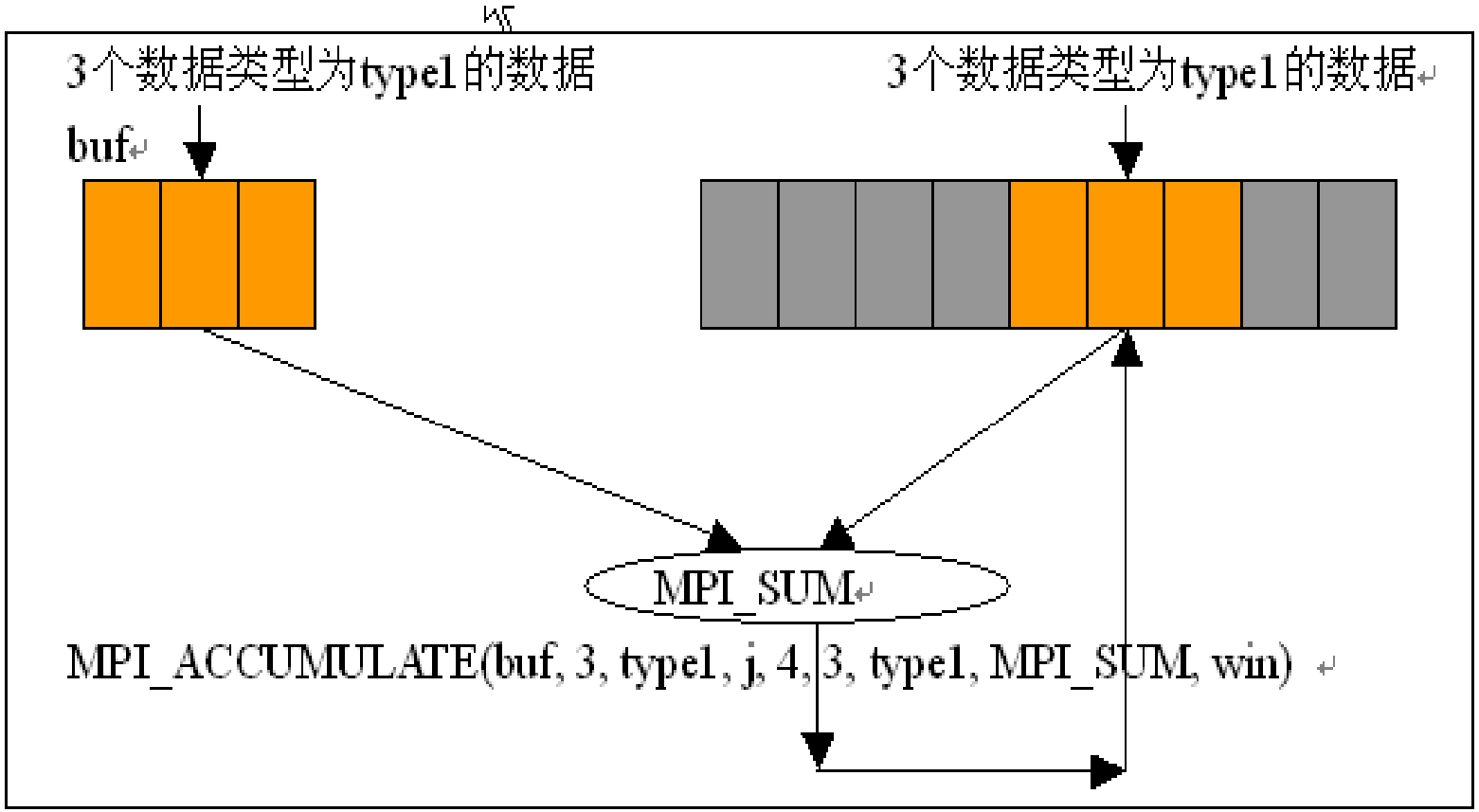
# 对远程数据的计算（窗口数据运算）

MPI_ACCUMULATE(origin_addr, origin_count, origin_datatype, target_rank, target_disp, target_count, target_datatype, op, win)



# 图示

- 同时执行了窗口的读和写操作



# 问题

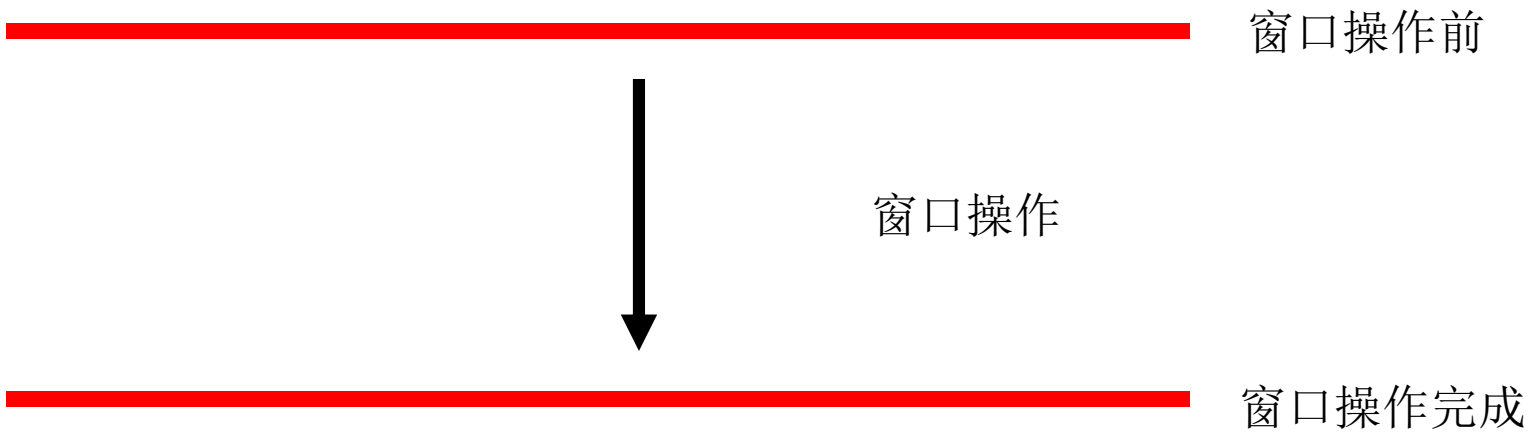
- 访问冲突问题（R,W）
  - 读读(RR)
  - 读写(RW)
  - 写读(WR)
  - 写写(WW)
- 解决办法
  - 窗口同步管理

# 窗口同步管理

- 栅栏方式
- 握手方式
- 锁方式

# 栅栏方式

- MPI_WIN_FENCE(assert, win)
- assert = 0



# 示例

| 进程0 | 进程1 | 进程N-1 |
| --- | --- | --- |
| MPI_WIN_FENCE | MPI_WIN_FENCE | MPI_WIN_FENCE |
| MPI_GET | | MPI_PUT |
| MPI_WIN_FENCE | MPI_WIN_FENCE | MPI_WIN_FENCE |

前后两次窗口操作互不干扰

# 握手方式

- 第一次握手，为窗口访问作准备
- 第二次握手，窗口访问完成

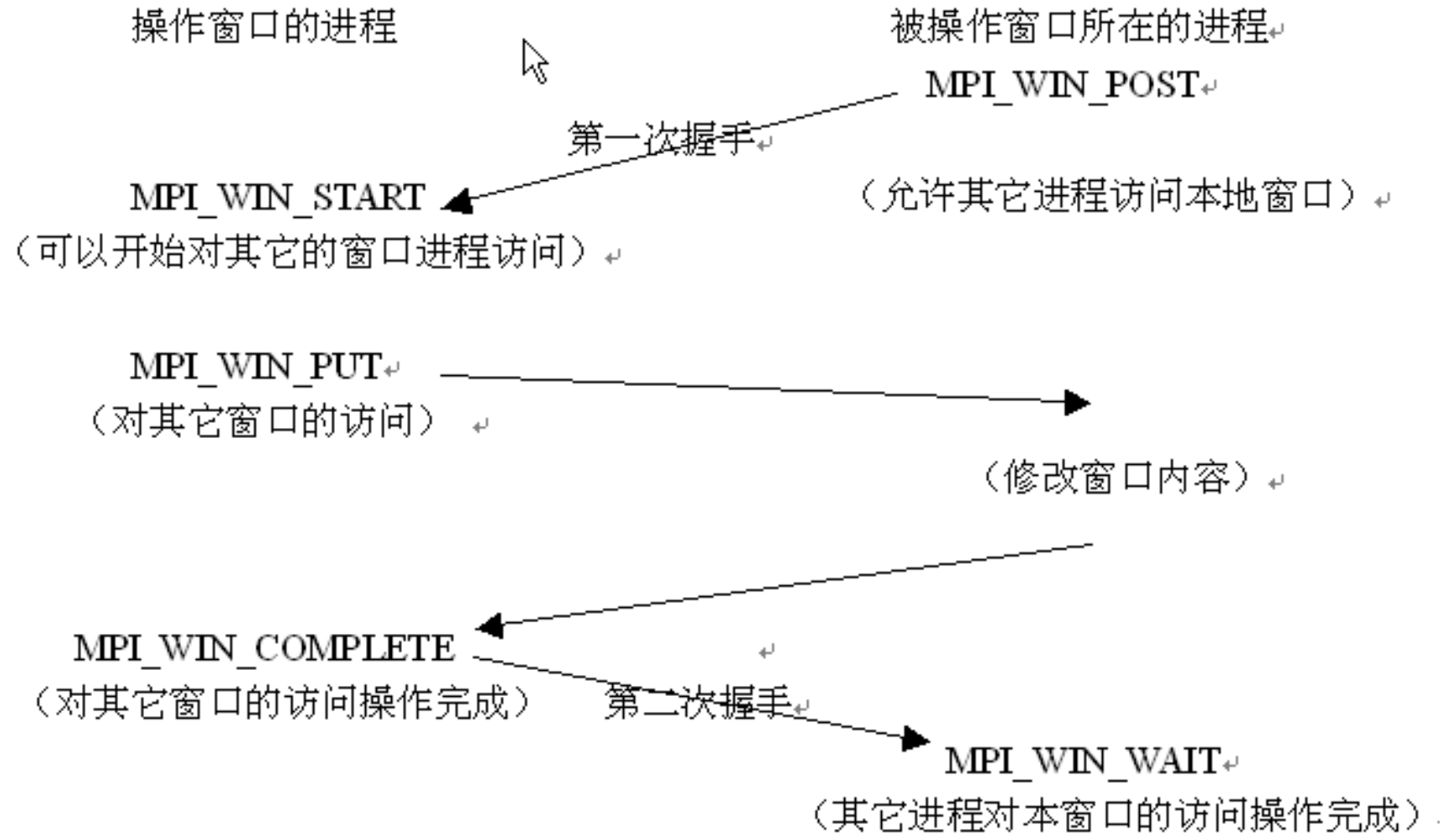
# 发起访问方的操作顺序

- MPI_WIN_START(group, assert, win) 启动第一次握手（开始窗口访问）
- 窗口操作
- MPI_WIN_COMPLETE(win) 启动第二次握手（结束窗口访问）

# 接收访问方的操作过程

- MPI_WIN_POST(group, assert, win) 启动第一次握手（准备窗口操作）
- 等待被访问
- MPI_WIN_WAIT(win) 启动第二次握手（窗口操作结束）

# 图示


操作窗口的进程
被操作窗口所在的进程
MPI_WIN_POST
第一次握手
MPI_WIN_START
（允许其它进程访问本地窗口）
（可以开始对其它的窗口进程访问）
MPI_WIN_PUT
（对其它窗口的访问）
（修改窗口内容）
MPI_WIN_COMPLETE
（对其它窗口的访问操作完成）
第二次握手
MPI_WIN_WAIT
（其它进程对本窗口的访问操作完成）

# 锁方式

- 借鉴临界区的概念
- 加锁后只允许自己访问
- 开锁后将访问权让给别人

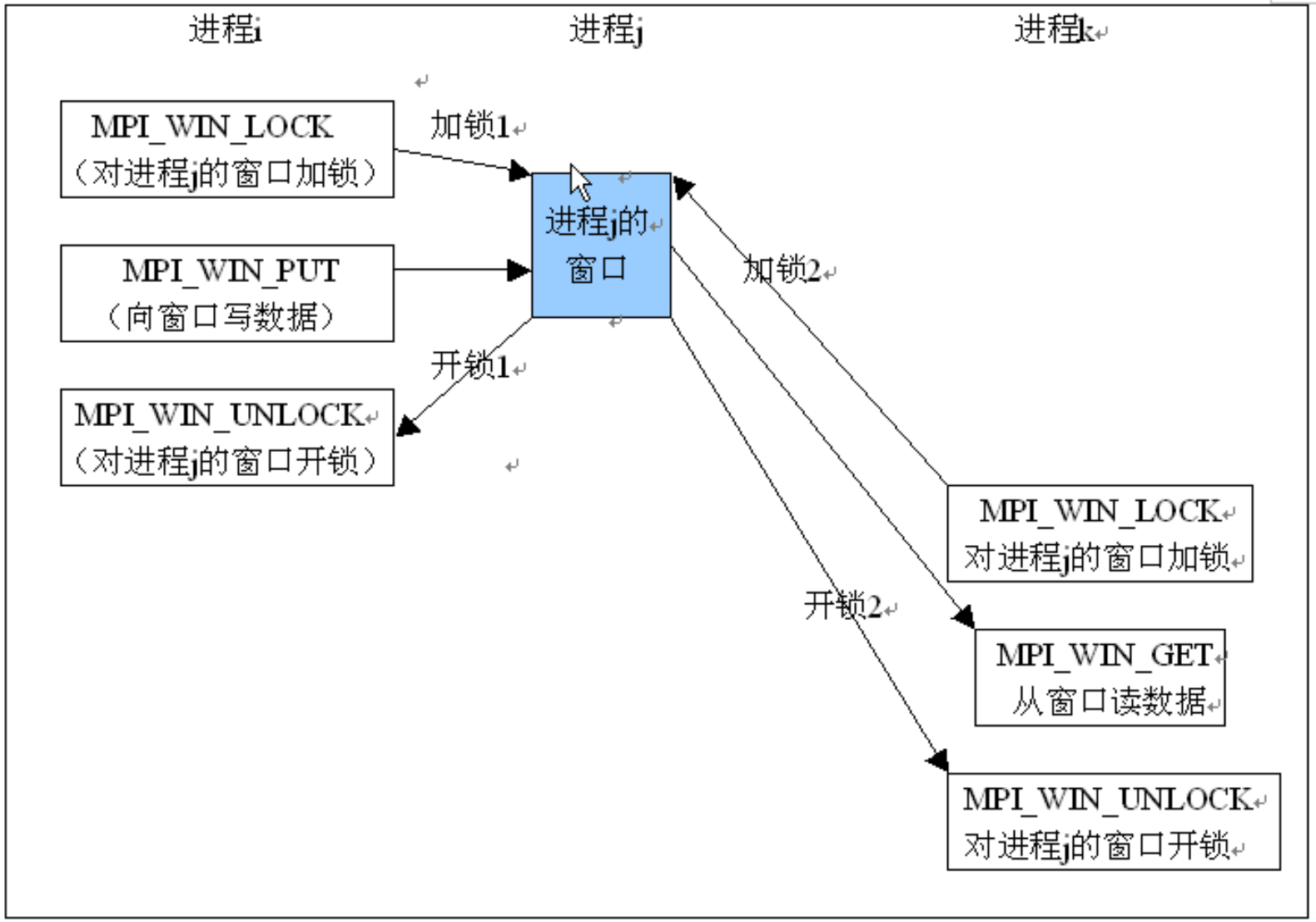
# 加锁语句

- MPI_WIN_LOCK(lock_type, rank, assert, win)
- 共享锁：允许其它的进程同时加共享锁
- 互斥锁：不允许其它的进程再加锁

# 开锁语句

- MPI_WIN_UNLOCK(rank, win)
- 允许其它的进程加锁

# 图示


进程i
进程j
进程k
MPI_WIN_LOCK
（对进程j的窗口加锁）
加锁1
进程j的
窗口
MPI_WIN_PUT
（向窗口写数据）
开锁1
MPI_WIN_UNLOCK
（对进程j的窗口开锁）
加锁2
MPI_WIN_LOCK
对进程j的窗口加锁
开锁2
MPI_WIN_GET
从窗口读数据
MPI_WIN_UNLOCK
对进程j的窗口开锁

# 练习

- 请用单边通信重新实现Jacobi迭代的例子。
- 请写一个简单的动态进行创建的例子，并实现父子进程之间的相互通信（相互发送并打印收到的内容）