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的并行文件I/O

都志辉
清华大学计算机系
Email ： duzh@tsinghua.edu.cn
Phone: 62782530

问题

用MPI-1的功能能否实现并行IO？

并行文件I/O的分类

- 显式偏移的文件I/O
- 视口文件I/O
- 共享文件I/O

一些基本操作

并行文件打开

- MPI_FILE_OPEN(comm, filename, amode, info, fh)
- 文件打开
- 组调用，amode必须都相同
- fh是一个组句柄（与WIN类似）
- 与用C/Fortran打开文件不同

文件打开方式

打开方式	含义
MPI_MODE_RDONLY	只读
MPI_MODE_RDWR	读写
MPI_MODE_WRONLY	只写
MPI_MODE_CREATE	若文件不存在则创建
MPI_MODE_EXCL	创建不存在的新文件，若存在则错
MPI_MODE_DELETE_ON_CLOSE	关闭时删除
MPI_MODE_UNIQUE_OPEN	不能并发打开
MPI_MODE_SEQUENTIAL	文件只能顺序存取
MPI_MODE_APPEND	追加方式打开，初始文件指针指向文件尾

文件关闭

- MPI_FILE_CLOSE(fh)
- 注意：组调用

删除指定的文件

- MPI_FILE_DELETE(filename, info)

文件大小

- MPI_FILE_SET_SIZE(fh,size)
- 组调用，所有的size都相同
- MPI_FILE_GET_SIZE(fh,size)

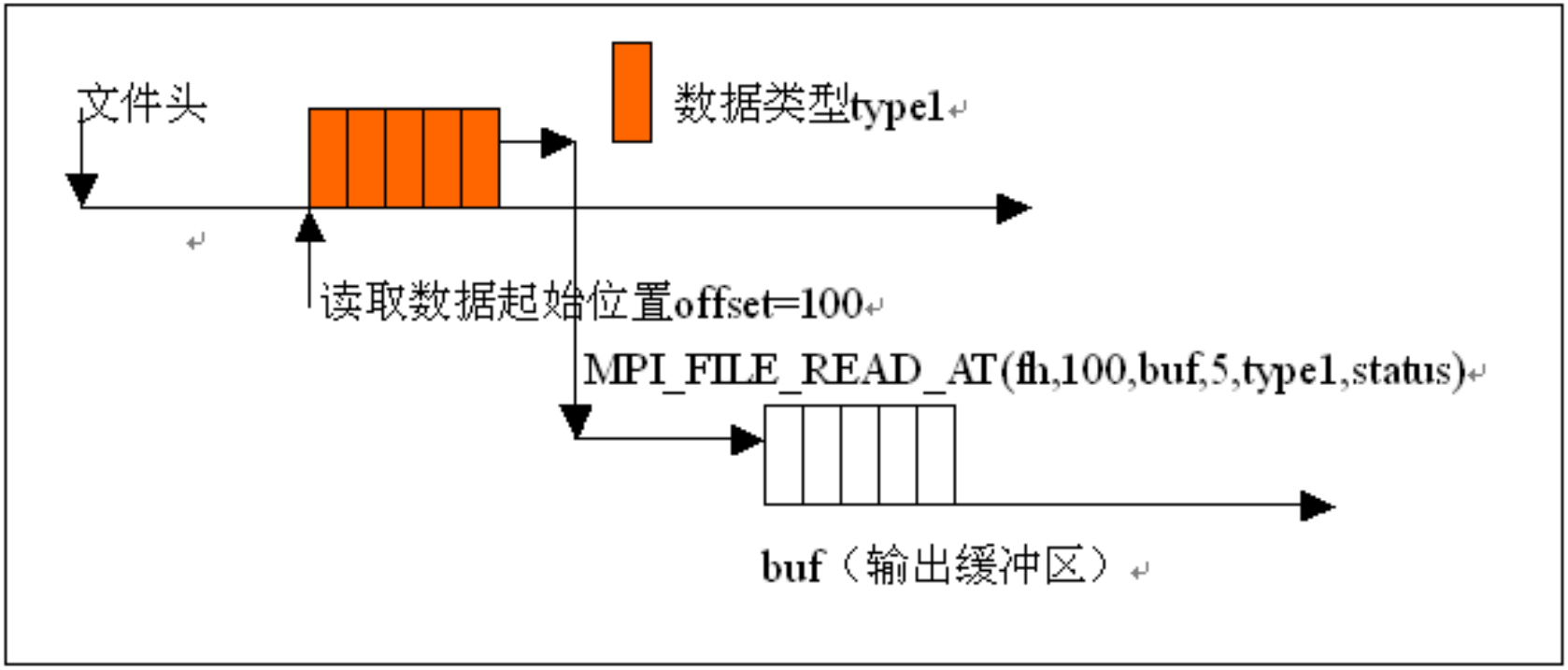
文件进程组与文件打开模式

- MPI_FILE_GET_GROUP(fh,group)
- 返回句柄fh对应的进程组group
- MPI_FILE_GET_AMODE(fh, amode)
- 返回打开文件时指定的模式

具有显式偏移的文件操作

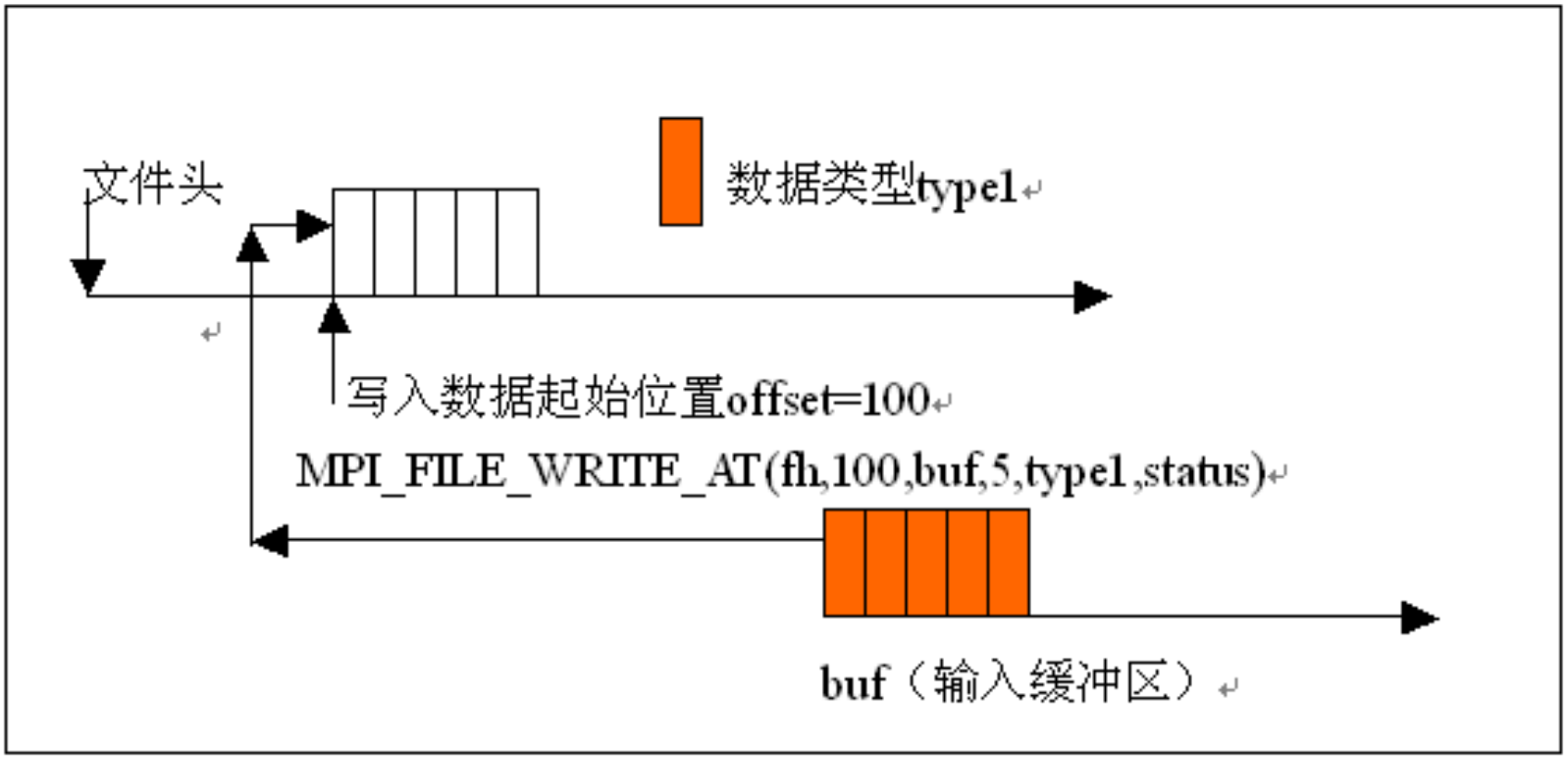
阻塞显式偏移并行文件读写

- MPI_FILE_READ_AT(fh, offset, buf, count, datatype,status)



写文件

- MPI_FILE_WRITE_AT(fh, offset, buf, count, datatype,status)



组调用阻塞显式偏移并行文件读写

- MPI_FILE_READ_AT_ALL(fh, offset,buf, count,datatype,status)
- 组调用，所有进程都执行了一次 MPI_FILE_READ_AT
- MPI_FILE_WRITE_AT_ALL(fh, offset,buf, count,datatype,status)
- 组调用，所有进程都执行了一次 MPI_FILE_WRITE_AT

非阻塞显式偏移并行文件读写

- MPI_FILE_IREAD_AT(fh, offset,buf, count, datatype, request)
- MPI_FILE_IWRITE_AT(fh, offset,buf, count, datatype, request)
- 同非阻塞通信的基本含义，调用返回并不意味着操作的完成

完成形式

- MPI_WAIT
- MPI_TEST
- 与非阻塞通信的完成调用形式完全相同

非阻塞组调用显式偏移并行文件读写

- 具有显式的开始与结束形式
- MPI_FILE_READ_AT_ALL_BEGIN(fh, offset, buf, count, datatype)
- 组调用读开始
- MPI_FILE_READ_AT_ALL_END(fh, buf, status)
- 组调用读结束

写操作

- MPI_FILE_WRITE_AT_ALL_BEGIN(fh, offset, buf, count, datatype)
- 组调用写开始
- MPI_FILE_WRITE_AT_ALL_END(fh, buf, status)
- 组调用写结束

视口文件读写

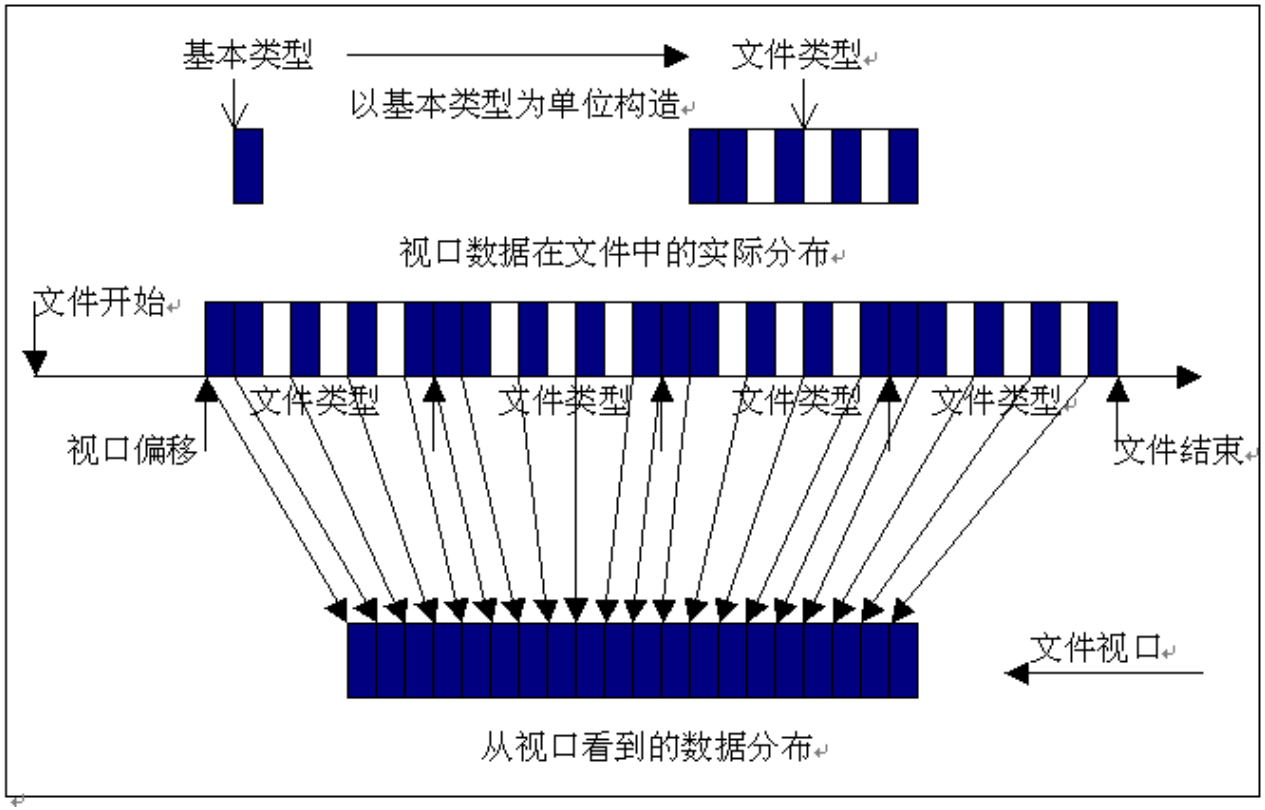
- 特点：隐式文件指针
- 取文件的一部分形成视口，然后对该视口进行操作，每个进程把视口看作是整个文件

文件视口

- <起始偏移，基本类型，文件类型>
- 文件类型是在基本类型的基础上形成的自定义数据类型

文件视口的定义

- MPI_FILE_SET_VIEW(fh, disp,etype, filetype, datarep,info)



数据表示

- native（效率最高，移植性最差）
- internal （效率一般，移植性一般）
- external32（效率最低，移植性最好）

文件视口的定义（续）

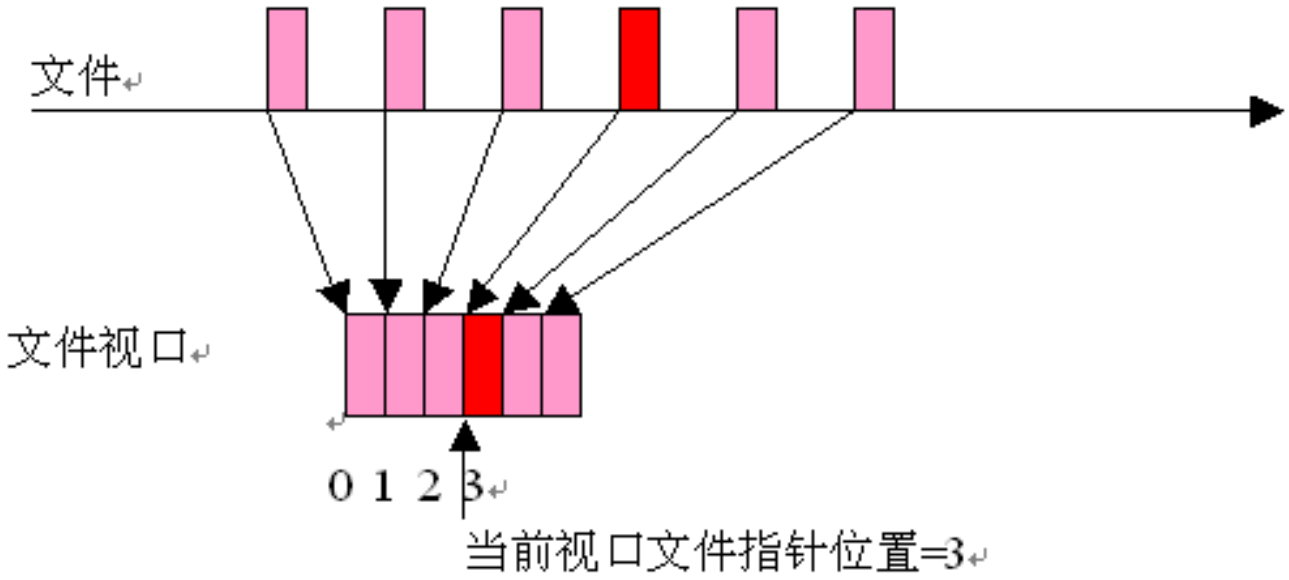
- 组调用
- 偏移的单位是字节
- 视口数据是连续的
- 文件句柄的含义转换

视口信息的获取

- MPI_FILE_GET_VIEW(fh, disp,etype, filetype, datarep)
- 得到偏移，基本类型，文件类型以及数据表示等信息

当前视口指针偏移

- MPI_FILE_GET_POSITION(fh, offset)
- 文件视口句柄，视口偏移



绝对偏移

- MPI_FILE_GET_BYTE_OFFSET(fh, offset, disp)
- 给定相对偏移offset，给出绝对偏移disp

阻塞方式的视口读写

- MPI_FILE_READ(fh, buf,count,datatype, status)
- 从视口当前位置读数据
- MPI_FILE_WRITE(fh, buf,count,datatype, status)
- 向视口当前位置写数据

阻塞方式的视口文件组调用

- MPI_FILE_READ_ALL(fh, buf,count, datatype,status)
- 组内所有进程都读
- MPI_FILE_WRITE_ALL(fh, buf,count, datatype,status)
- 组内所有进程都写

非阻塞视口文件读写

- MPI_FILE_IREAD(fh, buf, count, datatype, request)
- 非阻塞视口文件读
- MPI_FILE_IWRITE(fh, buf, count,datatype, request)
- 非阻塞视口文件写

完成操作

- MPI_WAIT
- MPI_TEST
- 同非阻塞的通信调用形式

非阻塞视口组调用

- MPI_FILE_READ_ALL_BEGIN(fh, buf, count,datatype)
- 组内进程都执行非阻塞的读操作
- MPI_FILE_READ_ALL_END(fh, buf, status)
- 组调用读操作完成

组调用写操作

- MPI_FILE_WRITE_ALL_BEGIN(fh, buf, count,datatype)
- 组调用写操作开始
- MPI_FILE_WRITE_ALL_END(fh, buf, status)
- 组调用写操作完成

共享文件操作

- 基于视口文件操作，但视口文件指针只有一个，即共享指针
- 任何文件对指针的操作都同时影响其它的文件
- 各个进程对视口定义有什么要求？

移动共享视口指针

- MPI_FILE_SEEK_SHARED(fh, offset, whence)

参照位置取值	调用后文件指针的位置
MPI_SEEK_SET	offset
MPI_SEEK_CUR	当前指针位置+offset
MPI_SEEK_END	文件结束位置+offset

阻塞共享文件读写

- MPI_FILE_READ_SHARED(fh, buf,count, datatype,status)
- 共享读
- MPI_FILE_WRITE_SHARED(fh, buf, count, datatype,status)
- 共享写

阻塞共享文件组读写

- MPI_FILE_READ_ORDERED(fh, buf, count, datatype, status)
- 组内进程依次读
- MPI_FILE_WRITE_ORDERED(fh, buf, count, datatype, status)
- 组内进程依次写

非阻塞共享指针文件操作

- MPI_FILE_IREAD_SHARED(fh, buf, count, datatype,request)
- 非阻塞读
- MPI_FILE_IWRITE_SHARED(fh, buf, count,datatype,request)
- 非阻塞写

完成调用

- MPI_WAIT
- MPI_TEST
- 同非阻塞通信形式

非阻塞共享文件组读写

- MPI_FILE_READ_ORDERED_BEGIN(fh, buf, count, datatype)
- 启动非阻塞组读操作
- MPI_FILE_READ_ORDERED_END(fh, buf, status)
- 完成非阻塞组读操作

写操作

- MPI_FILE_WRITE_ORDERED_BEGIN(fh, buf, count, datatype)
- 启动非阻塞组调用写操作
- MPI_FILE_WRITE_ORDERED_END(fh, buf, status)
- 完成非阻塞组调用写操作

分布式数组文件的存取

- MPI_TYPE_CREATE_DARRAY(size,rank,ndims,array_of_gsizes,array_of_distribs,array_of_dargs,array_of_psizes,order,oldtype,newtype)
- 定义一种新的数据类型，分布式数组文件类型

例子

```
gsizes[0]=m;
gzises[1]=n;
distribs[0]=MPI_DISTRIBUTE_BLOCK;        /*        MPI_DISTRIBUTE_CYCLIC,
MPI_DISTRIBUTE_NONE*/
distribs[1]=MPI_DISTRIBUTE_BLOCK;
dargs[0]=MPI_DISTRIBUTE_DFLT_DARG;
dargs[1]=MPI_DISTRIBUTE_DFLT_DARG;
psizes[0]=2;
psizes[1]=3;
MPI_Comm_rank(MPI_COMM_WORLD,&rank);
MPI_Type_create_darray(6,rank,2,gsizes,distribs,dargs,psizes,MPI_ORDER_C,MPI_
FLOAT, &filetype);
MPI_Type_commit(&filetype);
MPI_File_open(MPI_COMM_WORLD, "datafile",MPI_MODE_CREATE|
          MPI_MODE_WRONLY,MPI_INFO_NULL,&fh);
MPI_File_set_view(fh,0,MPI_FLOAT,filetype,"native",MPI_INFO_NULL);
local_array_size=num_local_rows*num_local_cols;
MPI_File_write_all(fh,local_array,local_array_size,MPI_FLOAT,&status);
MPI_File_close(&fh);
```

分布式子数组文件的定义

- MPI_TYPE_CREATE_SUBARRAY(ndims,array_of_sizes, array_of_subsizes, array_of_starts, order, oldtype, newtype)
- 在原来数组的基础上定义一个子数组类型

例子

```
gsizes[0]=m;
gsizes[1]=n;
psizes[0]=2;
psizes[3]=3;
lsizes[0]=m/psizes[0];
lsizes[1]=n/psizes[1];
dims[0]=2;
dims[1]=3;
periods[0]=periods[1]=1;
MPI_Cart_create(MPI_COMM_WORLD,2,dims,periods, 0 &comm);
MPI_Comm_rank(comm,&rank);
MPI_Cart_coords(comm,rank,2,coords);
start_indices[0]=coords[0]*lsizes[0];
start_indices[1]=coords[1]*lsizes[1];
```

例子

```
MPI_Type_create_subarray(2,gsizes,lsizes,start_indices,MPI_ORDER_C,MPI_
FLOAT,&filetype);
MPI_Type_commit(&filetype);
MPI_File_open(MPI_COMM_WORLD,"datafile",MPI_MODE_CREATE|
MPI_MODE_WRONLY, MPI_INFO_NULL, &fh);
MPI_File_setview(fh,0,MPI_FLOAT,filetype,"native",MPI_INFO_NULL);
memsizes[0]=lsizes[0]+8;
memsizes[1]=lsizes[1]+8;
/* 内存数组两侧都预留出4个单元*/
start_indices[0]=start_indices[1]=4;
MPI_Type_create_subarray(2,memsizes,lsizes,start_indices,MPI_ORDER_C,
MPI_FLOAT,&memtype);
/*定义内存数组的子数组*/
MPI_Type_commit(&memtype);
MPI_File_write_all(fh,local_array,1,memtype,&status);
MPI_File_close(&fh);
```

练习

- 1设进程个数为N，请定义一个文件类型，可以按照N间隔形成文件视口
- 2请定义一个二维的按照（block,*）形式分布的分布式数组类型，并将该分布式数组写到一个文件DA中。