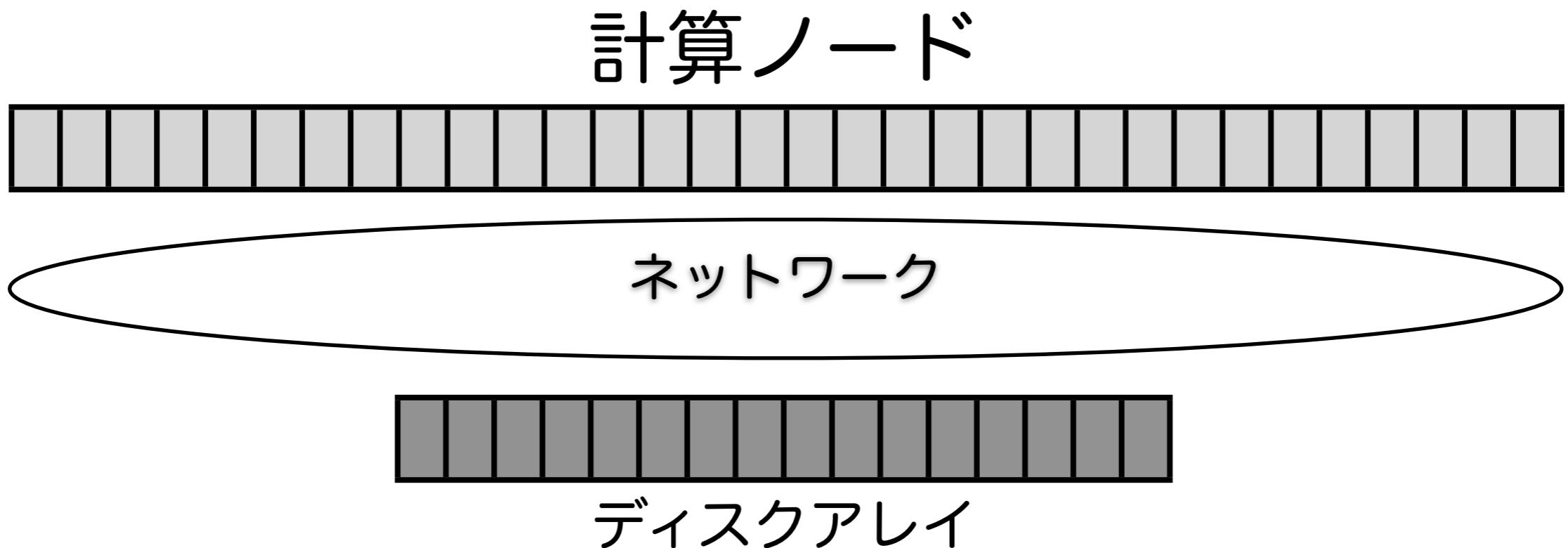


# MPI-IO

理化学研究所 AICS  
システムソフトウェア研究チーム  
堀 敦史

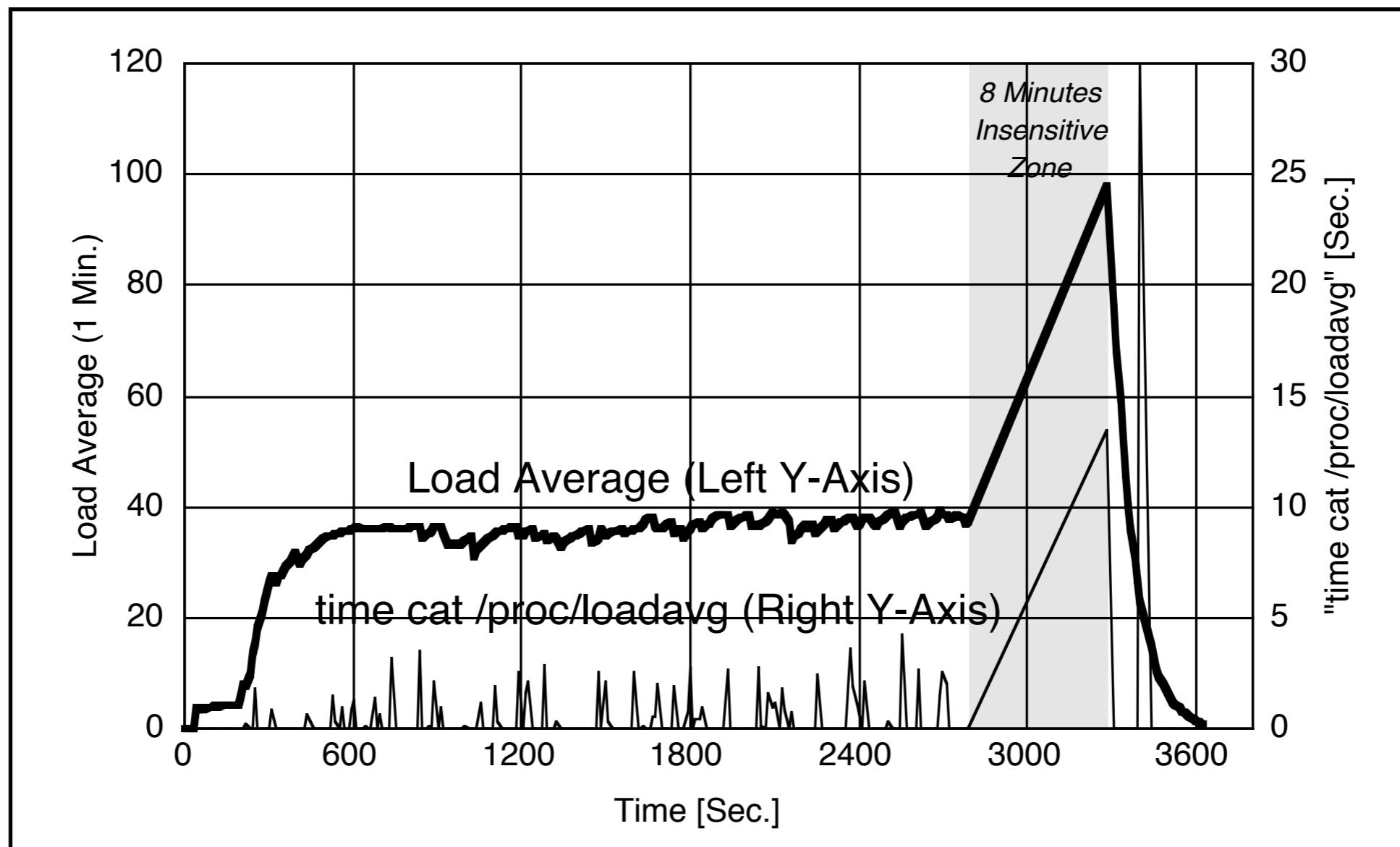
# 並列ファイルシステムの構成

- 並列ファイルシステムでは、ネットワークを経由してファイルにアクセスする
- 通信の特性として、小さいメッセージは遅いので、結果として、小さい単位でのファイル I/O は遅くなる



# 並列ファイルシステムの必要性

- よく使われている NFS では、まったく性能が出ない！



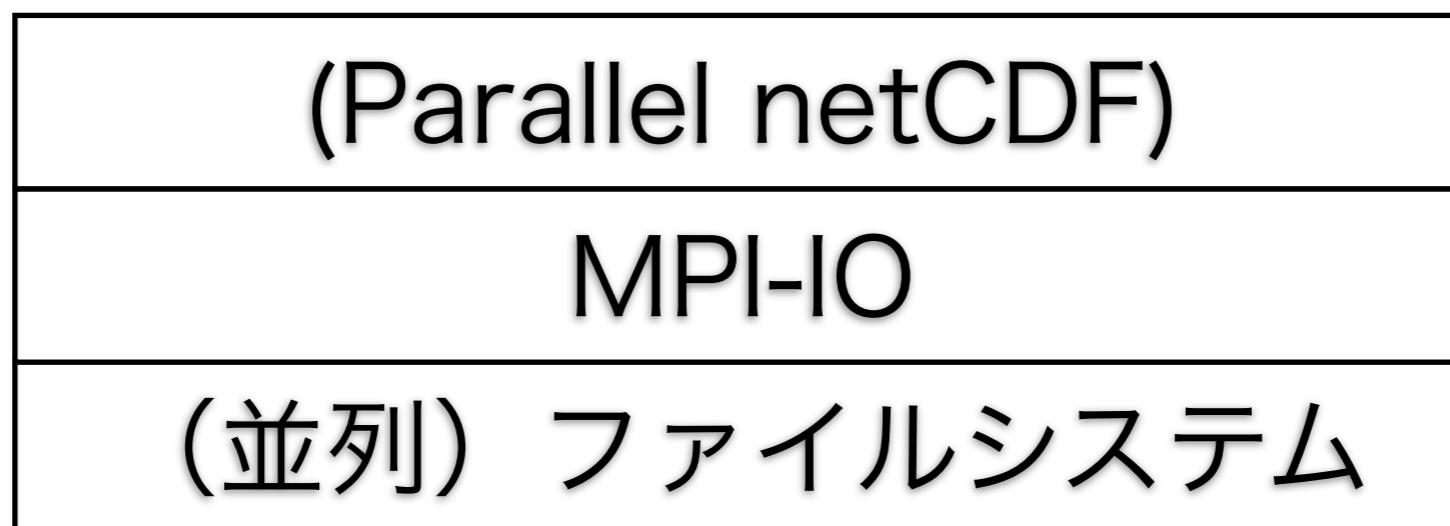
A timeline of NFS server load average when 64 processes (4 nodes) are creating 1GB file

# 並列ファイルシステム

- ・ 計算ノード数に（ほぼ）比例する IO バンド幅を提供する
- ・ 普通のディスクのバンド幅は 60 MB/s 前後
  - ・ 普通の SSD で 100 MB/s 前後
  - ・ これより大きなバンド幅は、複数のディスクを高速ネットワークで束ねて実現している
    - ・ **小さな単位のアクセスでは性能が全く出ない！**
    - ・ MPI-IO ではさまざまな工夫がある

# MPI-IO のソフトウェア構造

- MPI-IO は（並列）ファイルシステムを MPI という枠組みで抽象化したもの
- 性能向上のためにヒント情報を渡すことができるようになっている
  - MPI-IO の実装に依存したパラメータ
  - 機種やファイルシステム依存したパラメータ
- MPI-IO を用いた並列 IO ライブラリも存在する
  - Parallel netCDF <http://trac.mcs.anl.gov/projects/parallel-netcdf/>



# MPI-IO - open と close

```
C: MPI_File_open( MPI_Comm comm, char *filename,  
                  int amode, MPI_Info info, MPI_File *fh)  
    MPI_File_close( MPI_File *fh )  
F: MPI_FILE_OPEN( comm, filename, amode, info, fh, ierr )  
    MPI_FILE_CLOSE( fh, ierr )
```

- ・ 集団呼び出し（コミュニケーションオブジェクト comm に属する全てのプロセスで同じように呼び出す必要がある）
- ・ **amode** : 次頁参照
- ・ **info** : ファイルシステムへのヒント (MPI\_INFO\_NULL)
- ・ **fh** : ファイルハンドル - これに対しファイルを操作する

# MPI\_File\_open における amode

- amode - アクセスマード
  - 以下の値のビット OR

- MPI\_MODE\_RDONLY — read only,
- MPI\_MODE\_RDWR — reading and writing,
- MPI\_MODE\_WRONLY — write only,
- MPI\_MODE\_CREATE — create the file if it does not exist,
- MPI\_MODE\_EXCL — error if creating file that already exists,
- MPI\_MODE\_DELETE\_ON\_CLOSE — delete file on close,
- MPI\_MODE\_UNIQUE\_OPEN — file will not be concurrently opened elsewhere,
- MPI\_MODE\_SEQUENTIAL — file will only be accessed sequentially,
- MPI\_MODE\_APPEND — set initial position of all file pointers to end of file.

# MPI\_File\_set\_view

```
C: MPI_File_set_view( MPI_File *fh, MPI_Offset disp,  
                      MPI_Datatype etype, MPI_Datatype ftype,  
                      char *datarep )  
F: MPI_FILE_SET_VIEW( fh, disp, etype, ftype, datarep, ierr )
```

- Collective 関数
- **ftype** : etype の派生データ型で、これを呼んだプロセスがそのプロセスの **FP を用いて** アクセスするファイルの部分を示す
- **datarep** : ファイルに格納する数値データの表現方式 (互換性)
  - “native” マシン固有のバイナリー表現 (無変換)
  - “internal” (native と external32 の中間)
  - “external32” もっとも一般的で可搬性のあるバイナリー表現
- **info** : ファイルシステムに与えるヒント情報 (システム依存)

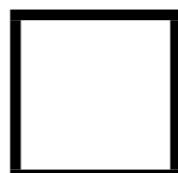
# MPI-IO における Datatype

- etype (elementary type)
  - ファイルにアクセスする場所を示す基本的な型
  - READ/WRITE で指定されるデータ型と type *signature* が同じでなければならない（各要素は同じ基本データ型であること）
- ftype (file type)
  - ファイルアクセスを分割するためのテンプレート
  - デフォルト
    - etype : MPI\_BYTE
    - ftype : MPI\_BYTE

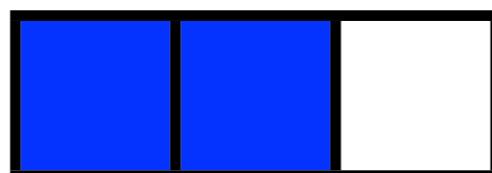
# `MPI_File_open` と `MPI_File_set_view`

- 通常 `MPI_File_set_view` は `MPI_File_open` の直後に宣言する

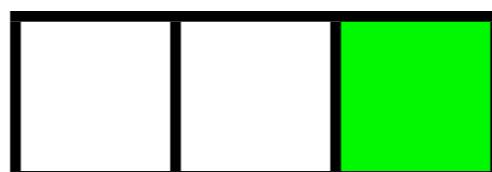
`etype:`



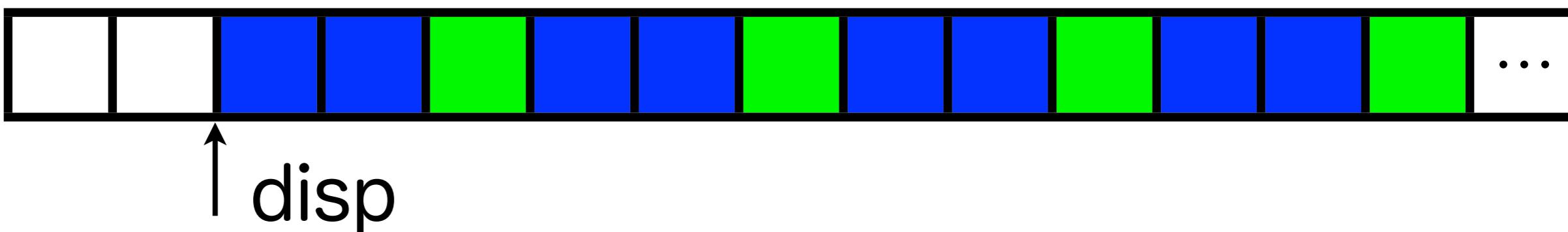
`ftype (proc 0):`



`ftype (proc 1):`



ファイル



# ファイルの IO

C: MPI\_File\_read( MPI\_File fh, void \*buf, int count, MPI\_Datatype type, MPI\_Status \*status)  
MPI\_File\_read\_all( MPI\_File fh, void \*buf, int count, MPI\_Datatype type, MPI\_Status \*status)  
MPI\_File\_read\_at( MPI\_File fh, MPI\_Offset off, void \*buf, int count, MPI\_Datatype type,  
MPI\_Status \*status)  
MPI\_File\_read\_at\_all( MPI\_File fh, MPI\_Offset off, void \*buf, int count, MPI\_Datatype type,  
MPI\_Status \*status)

MPI\_File\_write( MPI\_File fh, void \*buf, int count, MPI\_Datatype type, MPI\_Status \*status)  
MPI\_File\_write\_all( MPI\_File fh, void \*buf, int count, MPI\_Datatype type, MPI\_Status \*status)  
MPI\_File\_write\_at( MPI\_File fh, MPI\_Offset off, void \*buf, int count, MPI\_Datatype type,  
MPI\_Status \*status)  
MPI\_File\_write\_at\_all( MPI\_File fh, MPI\_Offset off, void \*buf, int count, MPI\_Datatype type,  
MPI\_Status \*status)

F: MPI\_FILE\_READ( fh, buf, count, datatype, status, ierr )  
MPI\_FILE\_READ\_ALL( fh, buf, count, datatype, status, ierr )  
MPI\_FILE\_READ\_AT( fh, offset, buf, count, datatype, status, ierr )  
MPI\_FILE\_READ\_AT\_ALL( fh, offset, buf, count, datatype, status, ierr )

MPI\_FILE\_WRITE( fh, buf, count, datatype, status, ierr )  
MPI\_FILE\_WRITE\_ALL( fh, buf, count, datatype, status, ierr )  
MPI\_FILE\_WRITE\_AT( fh, offset, buf, count, datatype, status, ierr )  
MPI\_FILE\_WRITE\_AT\_ALL( fh, offset, buf, count, datatype, status, ierr )

# MPI\_Status

C: MPI\_Get\_count( MPI\_Status status, MPI\_Datatype type, int \*count )  
F: MPI\_GET\_COUNT( status, type, count, ierr )

- MPI\_Status
  - 1対1通信における受信
  - MPI-IO におけるデータの read/write
  - MPI\_Status に対し、MPI\_GET\_COUNT を呼ぶと、その status を返した操作におけるデータの個数が返る
    - 1対1通信の受信：受信したデータの個数
    - MPI-IO : I/O したデータの個数

# ファイルアクセス関数の分類

positioning	synchronism	coordination	
		<i>noncollective</i>	<i>collective</i>
<i>explicit offsets</i>	<i>blocking</i>	MPI_FILE_READ_AT MPI_FILE_WRITE_AT	MPI_FILE_READ_AT_ALL MPI_FILE_WRITE_AT_ALL
	<i>nonblocking &amp; split collective</i>	MPI_FILE_IREAD_AT  MPI_FILE_IWRITE_AT	MPI_FILE_READ_AT_ALL_BEGIN MPI_FILE_READ_AT_ALL_END MPI_FILE_WRITE_AT_ALL_BEGIN MPI_FILE_WRITE_AT_ALL_END
<i>individual file pointers</i>	<i>blocking</i>	MPI_FILE_READ MPI_FILE_WRITE	MPI_FILE_READ_ALL MPI_FILE_WRITE_ALL
	<i>nonblocking &amp; split collective</i>	MPI_FILE_IREAD  MPI_FILE_IWRITE	MPI_FILE_READ_ALL_BEGIN MPI_FILE_READ_ALL_END MPI_FILE_WRITE_ALL_BEGIN MPI_FILE_WRITE_ALL_END
<i>shared file pointer</i>	<i>blocking</i>	MPI_FILE_READ_SHARED MPI_FILE_WRITE_SHARED	MPI_FILE_READ_ORDERED MPI_FILE_WRITE_ORDERED
	<i>nonblocking &amp; split collective</i>	MPI_FILE_IREAD_SHARED  MPI_FILE_IWRITE_SHARED	MPI_FILE_READ_ORDERED_BEGIN MPI_FILE_READ_ORDERED_END MPI_FILE_WRITE_ORDERED_BEGIN MPI_FILE_WRITE_ORDERED_END

# MPI-IO におけるファイルのアクセス

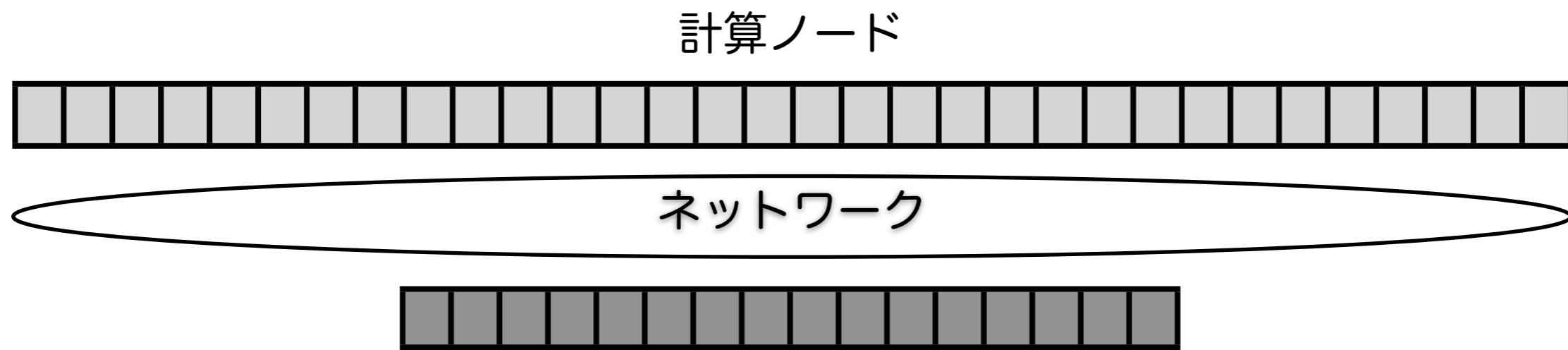
- Collective IO
  - 集団 IO
- File Pointer
  - アクセスする場所の指定
  - File View
    - 個々のプロセスへの割り付け
- (Non) Blocking IO
  - IO と計算のオーバーラップ

# ファイルアクセス 関数

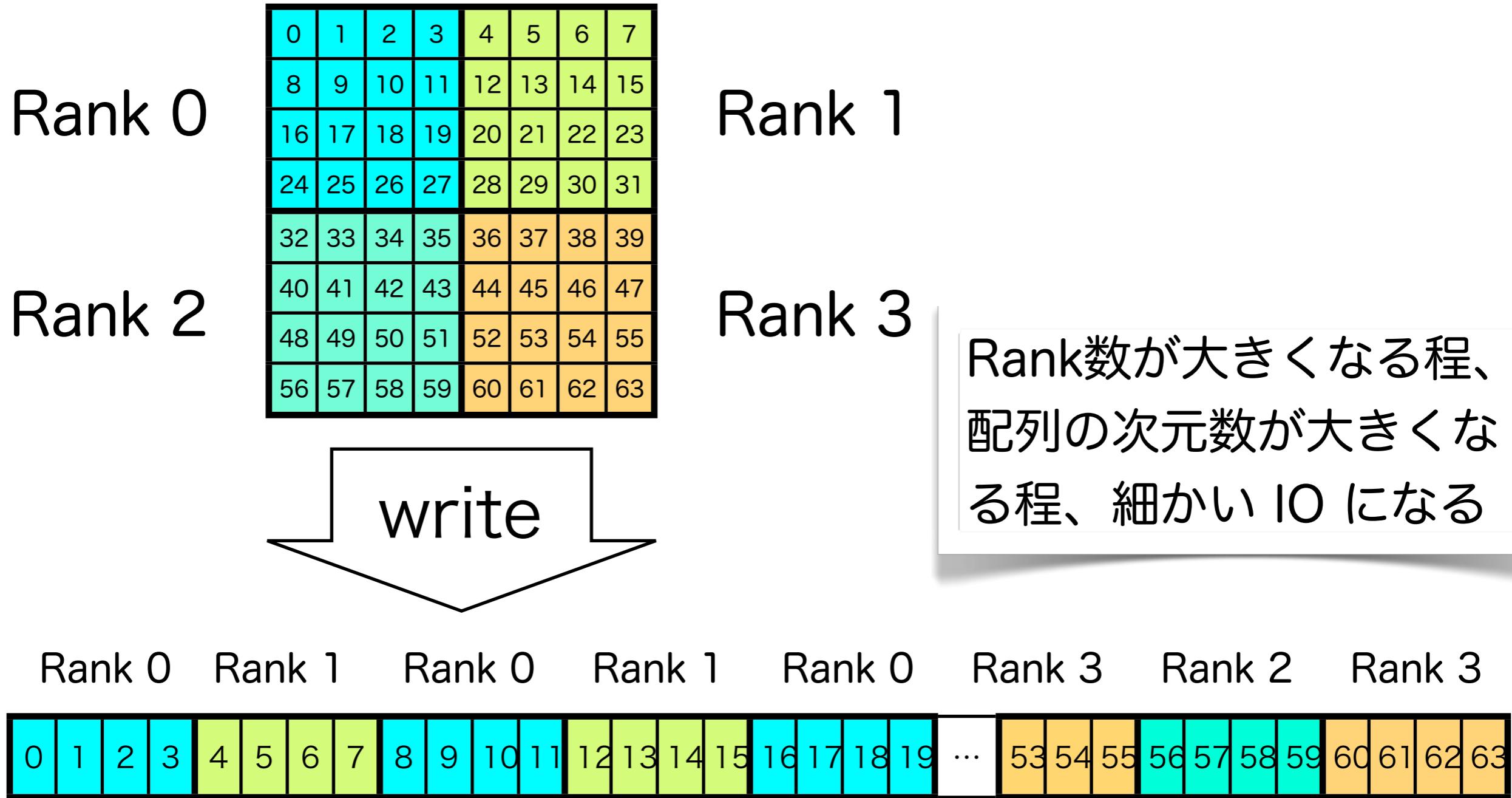
positioning	synchronism	coordination	
		<i>noncollective</i>	<i>collective</i>
<i>explicit offsets</i>	<i>blocking</i>	MPI_FILE_READ_AT MPI_FILE_WRITE_AT	MPI_FILE_READ_AT_ALL MPI_FILE_WRITE_AT_ALL
	<i>nonblocking &amp; split collective</i>	MPI_FILE_IREAD_AT  MPI_FILE_IWRITE_AT	MPI_FILE_READ_AT_ALL_BEGIN MPI_FILE_READ_AT_ALL_END MPI_FILE_WRITE_AT_ALL_BEGIN MPI_FILE_WRITE_AT_ALL_END
<i>individual file pointers</i>	<i>blocking</i>	MPI_FILE_READ MPI_FILE_WRITE	MPI_FILE_READ_ALL MPI_FILE_WRITE_ALL
	<i>nonblocking &amp; split collective</i>	MPI_FILE_IREAD  MPI_FILE_IWRITE	MPI_FILE_READ_ALL_BEGIN MPI_FILE_READ_ALL_END MPI_FILE_WRITE_ALL_BEGIN MPI_FILE_WRITE_ALL_END
<i>shared file pointer</i>	<i>blocking</i>	MPI_FILE_READ_SHARED MPI_FILE_WRITE_SHARED	MPI_FILE_READ_ORDERED MPI_FILE_WRITE_ORDERED
	<i>nonblocking &amp; split collective</i>	MPI_FILE_IREAD_SHARED  MPI_FILE_IWRITE_SHARED	MPI_FILE_READ_ORDERED_BEGIN MPI_FILE_READ_ORDERED_END MPI_FILE_WRITE_ORDERED_BEGIN MPI_FILE_WRITE_ORDERED_END

# 並列ファイルシステムの特性と並列プログラムの特性

- 並列ファイルシステムの特性
  - 複数のディスクを同時に使うことでバンド幅を稼ぐ
  - 複数のディスクを同時に使うほど、大きな単位でアクセスしないと、バンド幅がでない
- 並列プログラムの特性
  - **Weak Scaling** ノード数に比例して問題サイズが大きくなる
  - **Strong Scaling** ノード数に依らず問題サイズが一定
    - ノード数が大きくなるとノードあたりのデータは小さくなる
    - IO の性能が悪くなる

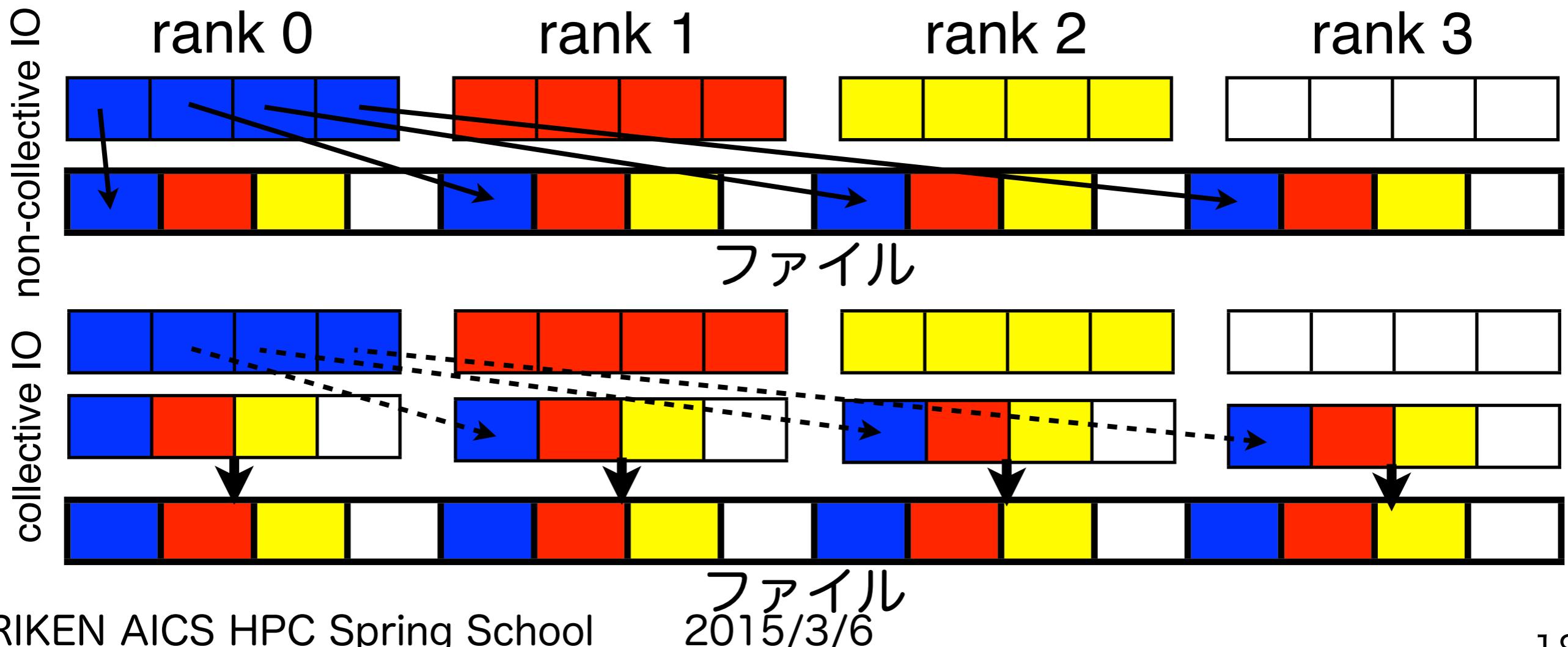


# 配列の block 分割の場合



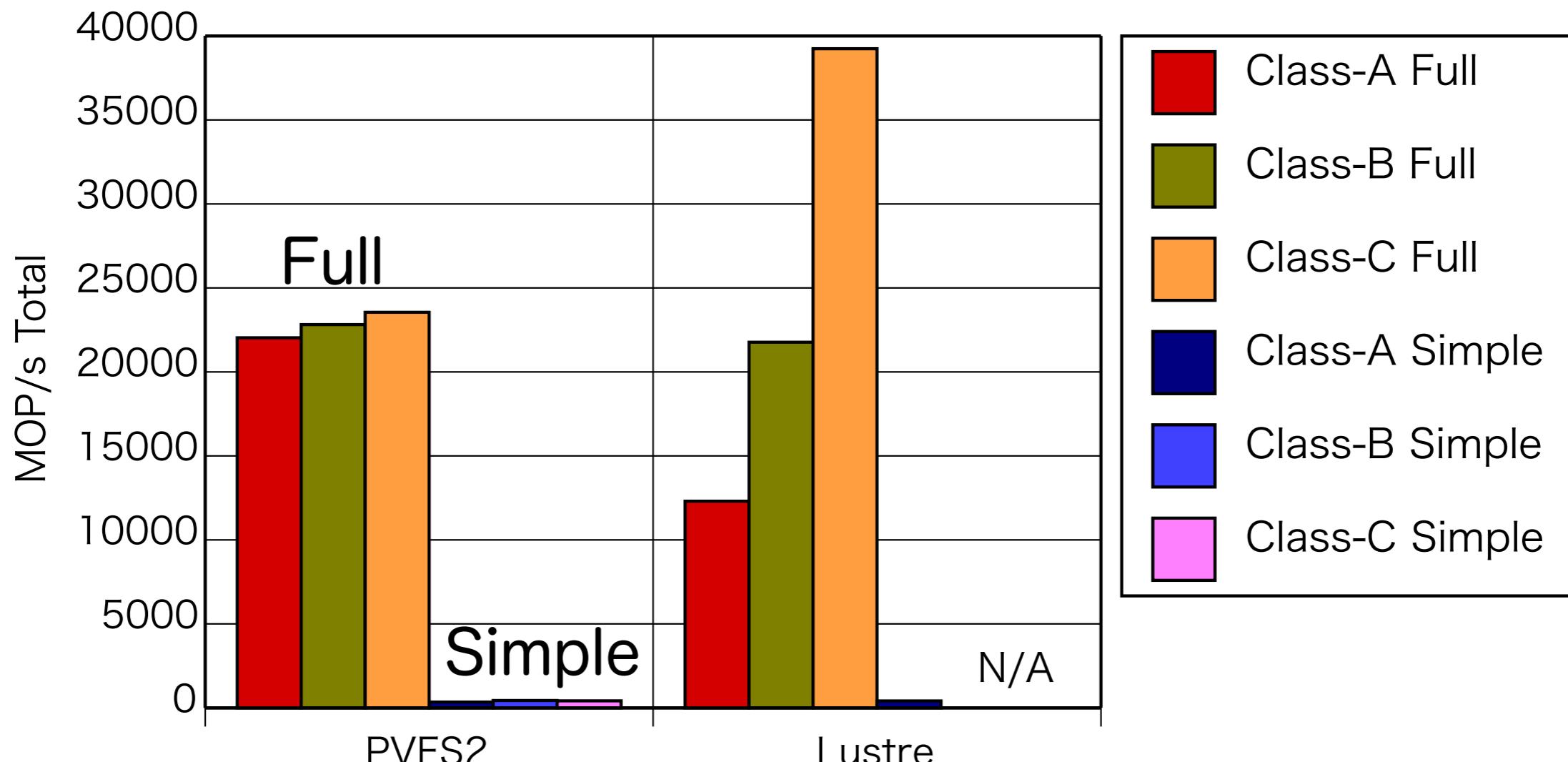
# Collective IO

- 全てのノードが同時に IO することで、細かいデータをとりまとめることができ（MPI の通信）、結果として IO のデータ量を大きくすることができる。Derived Datatype も同時に使い、1 回の IO 呼出でより多くの範囲のデータを指定することが大事
- 同時に、ファイルにアクセスする場所を連続にすることが可能
- 結果として、IO 性能が大きく向上する



# MPI-IO ベンチマーク

- BT-IO : MPI-IO ベンチマークプログラム
  - Class : 問題の大きさ ( $A < B < C$ )
  - Full : Collective IO
  - Simple : Non-Collective IO



# Collective I/O の問題点

## BT-IO Full

```

subroutine setup_btio
include 'header.h'
include 'mpinpb.h'

integer ierr
integer mstatus(MPI_STATUS_SIZE)
integer sizes(4), starts(4), subsizes(4)
integer cell_btype(maxcells), cell_ftype(maxcells)
integer cell_length(maxcells)
integer info
character*20 cb_nodes, cb_size
integer c, m
integer cell_disp(maxcells)

call mpi_bcast(collbuf_nodes, 1, MPI_INTEGER,
>           root, comm_setup, ierr)

call mpi_bcast(collbuf_size, 1, MPI_INTEGER,
>           root, comm_setup, ierr)

if (collbuf_nodes .eq. 0) then
  info = MPI_INFO_NULL
else
  write (cb_nodes,*) collbuf_nodes
  write (cb_size,*) collbuf_size
  call MPI_Info_create(info, ierr)
  call MPI_Info_set(info, 'cb_nodes', cb_nodes, ierr)
  call MPI_Info_set(info, 'cb_buffer_size', cb_size,
ierr)
  call MPI_Info_set(info, 'collective_buffering', 'true',
ierr)
endif

call MPI_Type_contiguous(5,
MPI_DOUBLE_PRECISION,
$           element, ierr)
call MPI_Type_commit(element, ierr)
call MPI_Type_extent(element, eltext, ierr)

do c = 1, ncells
c
c Outer array dimensions ar same for every cell
c
  sizes(1) = IMAX+4
  sizes(2) = JMAX+4
  sizes(3) = KMAX+4
c
c 4th dimension is cell number, total of maxcells cells
c
  sizes(4) = maxcells
c
c Internal dimensions of cells can differ slightly between
cells
c
  subsizes(1) = cell_size(1, c)
c
c Starting point in full array of c'th cell
c
  subsizes(2) = cell_size(2, c)
  subsizes(3) = cell_size(3, c)
c
c Cell is 4th dimension, 1 cell per cell type to handle
varying
c cell sub-array sizes
c
  subsizes(4) = 1
c
c type constructors use 0-based start addresses
c
  starts(1) = 2
  starts(2) = 2
  starts(3) = 2
  starts(4) = c-1
c
c Create buftype for a cell
c
  call MPI_Type_create_subarray(4, sizes,
subsizes,
$           starts, MPI_ORDER_FORTRAN, element,
$           cell_btype(c), ierr)
c
c block length and displacement for joining cells -
c 1 cell buftype per block, cell buftypes have own
displacement
c generated from cell number (4th array dimension)
c
  cell_length(c) = 1
  cell_disp(c) = 0
enddo

c
c Create combined buftype for all cells
c
  call MPI_Type_struct(ncells, cell_length, cell_disp,
$           cell_btype, combined_btype, ierr)
  call MPI_Type_commit(combined_btype, ierr)

  do c = 1, ncells
c
c Entire array size
c
    sizes(1) = PROBLEM_SIZE
    sizes(2) = PROBLEM_SIZE
    sizes(3) = PROBLEM_SIZE
c
c Size of c'th cell
c
    subsizes(1) = cell_size(1, c)
    subsizes(2) = cell_size(2, c)
    subsizes(3) = cell_size(3, c)
c
c Starting point in full array of c'th cell
c
  ends(1) = cell_low(1,c)
  ends(2) = cell_low(2,c)
  ends(3) = cell_low(3,c)
c
c Type of combined buftype
c
  starts(4) = 1
  starts(5) = 1
  starts(6) = 1
  starts(7) = 1
  starts(8) = 1
  starts(9) = 1
  starts(10) = 1
  starts(11) = 1
  starts(12) = 1
  starts(13) = 1
  starts(14) = 1
  starts(15) = 1
  starts(16) = 1
  starts(17) = 1
  starts(18) = 1
  starts(19) = 1
  starts(20) = 1
  starts(21) = 1
  starts(22) = 1
  starts(23) = 1
  starts(24) = 1
  starts(25) = 1
  starts(26) = 1
  starts(27) = 1
  starts(28) = 1
  starts(29) = 1
  starts(30) = 1
  starts(31) = 1
  starts(32) = 1
  starts(33) = 1
  starts(34) = 1
  starts(35) = 1
  starts(36) = 1
  starts(37) = 1
  starts(38) = 1
  starts(39) = 1
  starts(40) = 1
  starts(41) = 1
  starts(42) = 1
  starts(43) = 1
  starts(44) = 1
  starts(45) = 1
  starts(46) = 1
  starts(47) = 1
  starts(48) = 1
  starts(49) = 1
  starts(50) = 1
  starts(51) = 1
  starts(52) = 1
  starts(53) = 1
  starts(54) = 1
  starts(55) = 1
  starts(56) = 1
  starts(57) = 1
  starts(58) = 1
  starts(59) = 1
  starts(60) = 1
  starts(61) = 1
  starts(62) = 1
  starts(63) = 1
  starts(64) = 1
  starts(65) = 1
  starts(66) = 1
  starts(67) = 1
  starts(68) = 1
  starts(69) = 1
  starts(70) = 1
  starts(71) = 1
  starts(72) = 1
  starts(73) = 1
  starts(74) = 1
  starts(75) = 1
  starts(76) = 1
  starts(77) = 1
  starts(78) = 1
  starts(79) = 1
  starts(80) = 1
  starts(81) = 1
  starts(82) = 1
  starts(83) = 1
  starts(84) = 1
  starts(85) = 1
  starts(86) = 1
  starts(87) = 1
  starts(88) = 1
  starts(89) = 1
  starts(90) = 1
  starts(91) = 1
  starts(92) = 1
  starts(93) = 1
  starts(94) = 1
  starts(95) = 1
  starts(96) = 1
  starts(97) = 1
  starts(98) = 1
  starts(99) = 1
  starts(100) = 1
  starts(101) = 1
  starts(102) = 1
  starts(103) = 1
  starts(104) = 1
  starts(105) = 1
  starts(106) = 1
  starts(107) = 1
  starts(108) = 1
  starts(109) = 1
  starts(110) = 1
  starts(111) = 1
  starts(112) = 1
  starts(113) = 1
  starts(114) = 1
  starts(115) = 1
  starts(116) = 1
  starts(117) = 1
  starts(118) = 1
  starts(119) = 1
  starts(120) = 1
  starts(121) = 1
  starts(122) = 1
  starts(123) = 1
  starts(124) = 1
  starts(125) = 1
  starts(126) = 1
  starts(127) = 1
  starts(128) = 1
  starts(129) = 1
  starts(130) = 1
  starts(131) = 1
  starts(132) = 1
  starts(133) = 1
  starts(134) = 1
  starts(135) = 1
  starts(136) = 1
  starts(137) = 1
  starts(138) = 1
  starts(139) = 1
  starts(140) = 1
  starts(141) = 1
  starts(142) = 1
  starts(143) = 1
  starts(144) = 1
  starts(145) = 1
  starts(146) = 1
  starts(147) = 1
  starts(148) = 1
  starts(149) = 1
  starts(150) = 1
  starts(151) = 1
  starts(152) = 1
  starts(153) = 1
  starts(154) = 1
  starts(155) = 1
  starts(156) = 1
  starts(157) = 1
  starts(158) = 1
  starts(159) = 1
  starts(160) = 1
  starts(161) = 1
  starts(162) = 1
  starts(163) = 1
  starts(164) = 1
  starts(165) = 1
  starts(166) = 1
  starts(167) = 1
  starts(168) = 1
  starts(169) = 1
  starts(170) = 1
  starts(171) = 1
  starts(172) = 1
  starts(173) = 1
  starts(174) = 1
  starts(175) = 1
  starts(176) = 1
  starts(177) = 1
  starts(178) = 1
  starts(179) = 1
  starts(180) = 1
  starts(181) = 1
  starts(182) = 1
  starts(183) = 1
  starts(184) = 1
  starts(185) = 1
  starts(186) = 1
  starts(187) = 1
  starts(188) = 1
  starts(189) = 1
  starts(190) = 1
  starts(191) = 1
  starts(192) = 1
  starts(193) = 1
  starts(194) = 1
  starts(195) = 1
  starts(196) = 1
  starts(197) = 1
  starts(198) = 1
  starts(199) = 1
  starts(200) = 1
  starts(201) = 1
  starts(202) = 1
  starts(203) = 1
  starts(204) = 1
  starts(205) = 1
  starts(206) = 1
  starts(207) = 1
  starts(208) = 1
  starts(209) = 1
  starts(210) = 1
  starts(211) = 1
  starts(212) = 1
  starts(213) = 1
  starts(214) = 1
  starts(215) = 1
  starts(216) = 1
  starts(217) = 1
  starts(218) = 1
  starts(219) = 1
  starts(220) = 1
  starts(221) = 1
  starts(222) = 1
  starts(223) = 1
  starts(224) = 1
  starts(225) = 1
  starts(226) = 1
  starts(227) = 1
  starts(228) = 1
  starts(229) = 1
  starts(230) = 1
  starts(231) = 1
  starts(232) = 1
  starts(233) = 1
  starts(234) = 1
  starts(235) = 1
  starts(236) = 1
  starts(237) = 1
  starts(238) = 1
  starts(239) = 1
  starts(240) = 1
  starts(241) = 1
  starts(242) = 1
  starts(243) = 1
  starts(244) = 1
  starts(245) = 1
  starts(246) = 1
  starts(247) = 1
  starts(248) = 1
  starts(249) = 1
  starts(250) = 1
  starts(251) = 1
  starts(252) = 1
  starts(253) = 1
  starts(254) = 1
  starts(255) = 1
  starts(256) = 1
  starts(257) = 1
  starts(258) = 1
  starts(259) = 1
  starts(260) = 1
  starts(261) = 1
  starts(262) = 1
  starts(263) = 1
  starts(264) = 1
  starts(265) = 1
  starts(266) = 1
  starts(267) = 1
  starts(268) = 1
  starts(269) = 1
  starts(270) = 1
  starts(271) = 1
  starts(272) = 1
  starts(273) = 1
  starts(274) = 1
  starts(275) = 1
  starts(276) = 1
  starts(277) = 1
  starts(278) = 1
  starts(279) = 1
  starts(280) = 1
  starts(281) = 1
  starts(282) = 1
  starts(283) = 1
  starts(284) = 1
  starts(285) = 1
  starts(286) = 1
  starts(287) = 1
  starts(288) = 1
  starts(289) = 1
  starts(290) = 1
  starts(291) = 1
  starts(292) = 1
  starts(293) = 1
  starts(294) = 1
  starts(295) = 1
  starts(296) = 1
  starts(297) = 1
  starts(298) = 1
  starts(299) = 1
  starts(300) = 1
  starts(301) = 1
  starts(302) = 1
  starts(303) = 1
  starts(304) = 1
  starts(305) = 1
  starts(306) = 1
  starts(307) = 1
  starts(308) = 1
  starts(309) = 1
  starts(310) = 1
  starts(311) = 1
  starts(312) = 1
  starts(313) = 1
  starts(314) = 1
  starts(315) = 1
  starts(316) = 1
  starts(317) = 1
  starts(318) = 1
  starts(319) = 1
  starts(320) = 1
  starts(321) = 1
  starts(322) = 1
  starts(323) = 1
  starts(324) = 1
  starts(325) = 1
  starts(326) = 1
  starts(327) = 1
  starts(328) = 1
  starts(329) = 1
  starts(330) = 1
  starts(331) = 1
  starts(332) = 1
  starts(333) = 1
  starts(334) = 1
  starts(335) = 1
  starts(336) = 1
  starts(337) = 1
  starts(338) = 1
  starts(339) = 1
  starts(340) = 1
  starts(341) = 1
  starts(342) = 1
  starts(343) = 1
  starts(344) = 1
  starts(345) = 1
  starts(346) = 1
  starts(347) = 1
  starts(348) = 1
  starts(349) = 1
  starts(350) = 1
  starts(351) = 1
  starts(352) = 1
  starts(353) = 1
  starts(354) = 1
  starts(355) = 1
  starts(356) = 1
  starts(357) = 1
  starts(358) = 1
  starts(359) = 1
  starts(360) = 1
  starts(361) = 1
  starts(362) = 1
  starts(363) = 1
  starts(364) = 1
  starts(365) = 1
  starts(366) = 1
  starts(367) = 1
  starts(368) = 1
  starts(369) = 1
  starts(370) = 1
  starts(371) = 1
  starts(372) = 1
  starts(373) = 1
  starts(374) = 1
  starts(375) = 1
  starts(376) = 1
  starts(377) = 1
  starts(378) = 1
  starts(379) = 1
  starts(380) = 1
  starts(381) = 1
  starts(382) = 1
  starts(383) = 1
  starts(384) = 1
  starts(385) = 1
  starts(386) = 1
  starts(387) = 1
  starts(388) = 1
  starts(389) = 1
  starts(390) = 1
  starts(391) = 1
  starts(392) = 1
  starts(393) = 1
  starts(394) = 1
  starts(395) = 1
  starts(396) = 1
  starts(397) = 1
  starts(398) = 1
  starts(399) = 1
  starts(400) = 1
  starts(401) = 1
  starts(402) = 1
  starts(403) = 1
  starts(404) = 1
  starts(405) = 1
  starts(406) = 1
  starts(407) = 1
  starts(408) = 1
  starts(409) = 1
  starts(410) = 1
  starts(411) = 1
  starts(412) = 1
  starts(413) = 1
  starts(414) = 1
  starts(415) = 1
  starts(416) = 1
  starts(417) = 1
  starts(418) = 1
  starts(419) = 1
  starts(420) = 1
  starts(421) = 1
  starts(422) = 1
  starts(423) = 1
  starts(424) = 1
  starts(425) = 1
  starts(426) = 1
  starts(427) = 1
  starts(428) = 1
  starts(429) = 1
  starts(430) = 1
  starts(431) = 1
  starts(432) = 1
  starts(433) = 1
  starts(434) = 1
  starts(435) = 1
  starts(436) = 1
  starts(437) = 1
  starts(438) = 1
  starts(439) = 1
  starts(440) = 1
  starts(441) = 1
  starts(442) = 1
  starts(443) = 1
  starts(444) = 1
  starts(445) = 1
  starts(446) = 1
  starts(447) = 1
  starts(448) = 1
  starts(449) = 1
  starts(450) = 1
  starts(451) = 1
  starts(452) = 1
  starts(453) = 1
  starts(454) = 1
  starts(455) = 1
  starts(456) = 1
  starts(457) = 1
  starts(458) = 1
  starts(459) = 1
  starts(460) = 1
  starts(461) = 1
  starts(462) = 1
  starts(463) = 1
  starts(464) = 1
  starts(465) = 1
  starts(466) = 1
  starts(467) = 1
  starts(468) = 1
  starts(469) = 1
  starts(470) = 1
  starts(471) = 1
  starts(472) = 1
  starts(473) = 1
  starts(474) = 1
  starts(475) = 1
  starts(476) = 1
  starts(477) = 1
  starts(478) = 1
  starts(479) = 1
  starts(480) = 1
  starts(481) = 1
  starts(482) = 1
  starts(483) = 1
  starts(484) = 1
  starts(485) = 1
  starts(486) = 1
  starts(487) = 1
  starts(488) = 1
  starts(489) = 1
  starts(490) = 1
  starts(491) = 1
  starts(492) = 1
  starts(493) = 1
  starts(494) = 1
  starts(495) = 1
  starts(496) = 1
  starts(497) = 1
  starts(498) = 1
  starts(499) = 1
  starts(500) = 1
  starts(501) = 1
  starts(502) = 1
  starts(503) = 1
  starts(504) = 1
  starts(505) = 1
  starts(506) = 1
  starts(507) = 1
  starts(508) = 1
  starts(509) = 1
  starts(510) = 1
  starts(511) = 1
  starts(512) = 1
  starts(513) = 1
  starts(514) = 1
  starts(515) = 1
  starts(516) = 1
  starts(517) = 1
  starts(518) = 1
  starts(519) = 1
  starts(520) = 1
  starts(521) = 1
  starts(522) = 1
  starts(523) = 1
  starts(524) = 1
  starts(525) = 1
  starts(526) = 1
  starts(527) = 1
  starts(528) = 1
  starts(529) = 1
  starts(530) = 1
  starts(531) = 1
  starts(532) = 1
  starts(533) = 1
  starts(534) = 1
  starts(535) = 1
  starts(536) = 1
  starts(537) = 1
  starts(538) = 1
  starts(539) = 1
  starts(540) = 1
  starts(541) = 1
  starts(542) = 1
  starts(543) = 1
  starts(544) = 1
  starts(545) = 1
  starts(546) = 1
  starts(547) = 1
  starts(548) = 1
  starts(549) = 1
  starts(550) = 1
  starts(551) = 1
  starts(552) = 1
  starts(553) = 1
  starts(554) = 1
  starts(555) = 1
  starts(556) = 1
  starts(557) = 1
  starts(558) = 1
  starts(559) = 1
  starts(560) = 1
  starts(561) = 1
  starts(562) = 1
  starts(563) = 1
  starts(564) = 1
  starts(565) = 1
  starts(566) = 1
  starts(567) = 1
  starts(568) = 1
  starts(569) = 1
  starts(570) = 1
  starts(571) = 1
  starts(572) = 1
  starts(573) = 1
  starts(574) = 1
  starts(575) = 1
  starts(576) = 1
  starts(577) = 1
  starts(578) = 1
  starts(579) = 1
  starts(580) = 1
  starts(581) = 1
  starts(582) = 1
  starts(583) = 1
  starts(584) = 1
  starts(585) = 1
  starts(586) = 1
  starts(587) = 1
  starts(588) = 1
  starts(589) = 1
  starts(590) = 1
  starts(591) = 1
  starts(592) = 1
  starts(593) = 1
  starts(594) = 1
  starts(595) = 1
  starts(596) = 1
  starts(597) = 1
  starts(598) = 1
  starts(599) = 1
  starts(600) = 1
  starts(601) = 1
  starts(602) = 1
  starts(603) = 1
  starts(604) = 1
  starts(605) = 1
  starts(606) = 1
  starts(607) = 1
  starts(608) = 1
  starts(609) = 1
  starts(610) = 1
  starts(611) = 1
  starts(612) = 1
  starts(613) = 1
  starts(614) = 1
  starts(615) = 1
  starts(616) = 1
  starts(617) = 1
  starts(618) = 1
  starts(619) = 1
  starts(620) = 1
  starts(621) = 1
  starts(622) = 1
  starts(623) = 1
  starts(624) = 1
  starts(625) = 1
  starts(626) = 1
  starts(627) = 1
  starts(628) = 1
  starts(629) = 1
  starts(630) = 1
  starts(631) = 1
  starts(632) = 1
  starts(633) = 1
  starts(634) = 1
  starts(635) = 1
  starts(636) = 1
  starts(637) = 1
  starts(638) = 1
  starts(639) = 1
  starts(640) = 1
  starts(641) = 1
  starts(642) = 1
  starts(643) = 1
  starts(644) = 1
  starts(645) = 1
  starts(646) = 1
  starts(647) = 1
  starts(648) = 1
  starts(649) = 1
  starts(650) = 1
  starts(651) = 1
  starts(652) = 1
  starts(653) = 1
  starts(654) = 1
  starts(655) = 1
  starts(656) = 1
  starts(657) = 1
  starts(658) = 1
  starts(659) = 1
  starts(660) = 1
  starts(661) = 1
  starts(662) = 1
  starts(663) = 1
  starts(664) = 1
  starts(665) = 1
  starts(666) = 1
  starts(667) = 1
  starts(668) = 1
  starts(669) = 1
  starts(670) = 1
  starts(671) = 1
  starts(672) = 1
  starts(673) = 1
  starts(674) = 1
  starts(675) = 1
  starts(676) = 1
  starts(677) = 1
  starts(678) = 1
  starts(679) = 1
  starts(680) = 1
  starts(681) = 1
  starts(682) = 1
  starts(683) = 1
  starts(684) = 1
  starts(685) = 1
  starts(686) = 1
  starts(687) = 1
  starts(
```

# コミュニケータを指定しない Collective な操作

- ・片方向通信の Window や MPI-IO のファイルハンドル
- ・それらの生成時にコミュニケータを指定しているから
  - ・生成時に指定されたコミュニケータを MPI\_COMM\_DUP して内部に持っている

# ファイルアクセス 関数

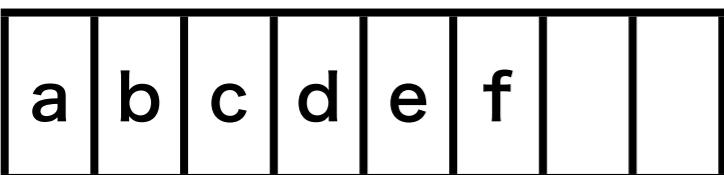
positioning	synchronism	coordination	
		<i>noncollective</i>	<i>collective</i>
<i>explicit offsets</i>	<i>blocking</i>	MPI_FILE_READ_AT MPI_FILE_WRITE_AT	MPI_FILE_READ_AT_ALL MPI_FILE_WRITE_AT_ALL
	<i>nonblocking &amp; split collective</i>	MPI_FILE_IREAD_AT  MPI_FILE_IWRITE_AT	MPI_FILE_READ_AT_ALL_BEGIN MPI_FILE_READ_AT_ALL_END MPI_FILE_WRITE_AT_ALL_BEGIN MPI_FILE_WRITE_AT_ALL_END
<i>individual file pointers</i>	<i>blocking</i>	MPI_FILE_READ MPI_FILE_WRITE	MPI_FILE_READ_ALL MPI_FILE_WRITE_ALL
	<i>nonblocking &amp; split collective</i>	MPI_FILE_IREAD  MPI_FILE_IWRITE	MPI_FILE_READ_ALL_BEGIN MPI_FILE_READ_ALL_END MPI_FILE_WRITE_ALL_BEGIN MPI_FILE_WRITE_ALL_END
<i>shared file pointer</i>	<i>blocking</i>	MPI_FILE_READ_SHARED MPI_FILE_WRITE_SHARED	MPI_FILE_READ_ORDERED MPI_FILE_WRITE_ORDERED
	<i>nonblocking &amp; split collective</i>	MPI_FILE_IREAD_SHARED  MPI_FILE_IWRITE_SHARED	MPI_FILE_READ_ORDERED_BEGIN MPI_FILE_READ_ORDERED_END MPI_FILE_WRITE_ORDERED_BEGIN MPI_FILE_WRITE_ORDERED_END

# File Pointer 問題

- 逐次プログラムの IO では、file pointer (アクセスする位置) がひとつだけ (暗黙に) ある
- 並列プログラムでは、複数のプロセスが同時に書込むために、file pointer の扱いに注意する必要がある

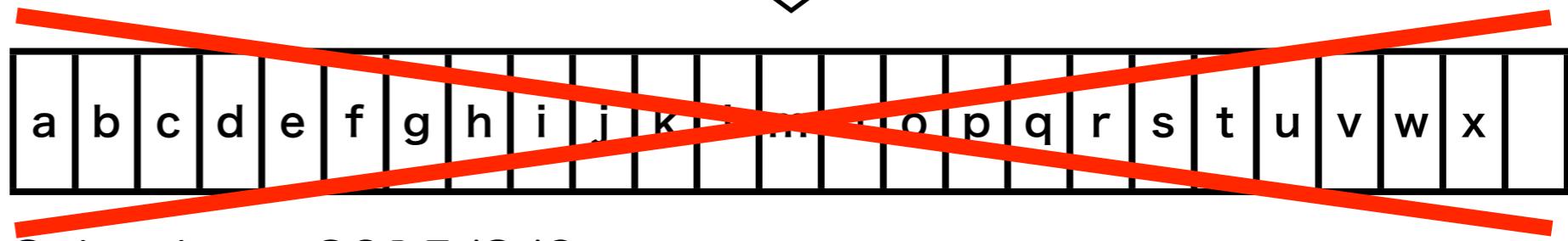
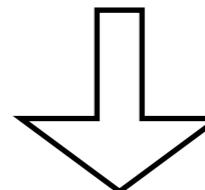
逐次

```
write( "abc" )  
write( "def" )
```



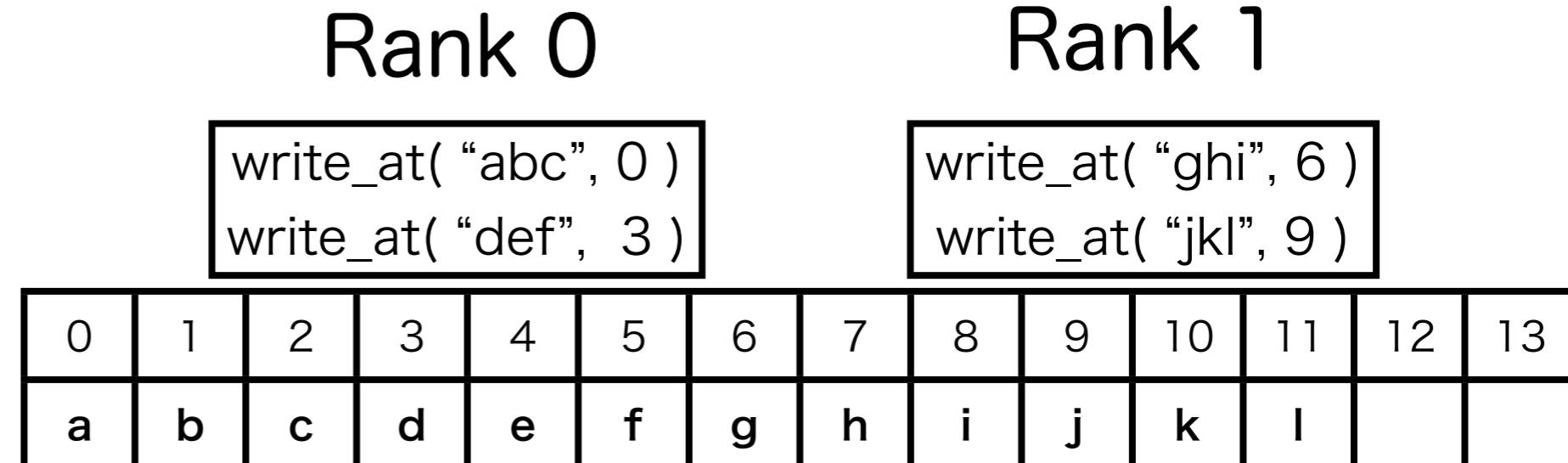
並列

Rank 0	Rank 1	Rank 2	Rank 3
write( "abc" ) write( "def" )	write( "ghi" ) write( "jkl" )	write( "mno" ) write( "pqr" )	write( "stu" ) write( "vwx" )

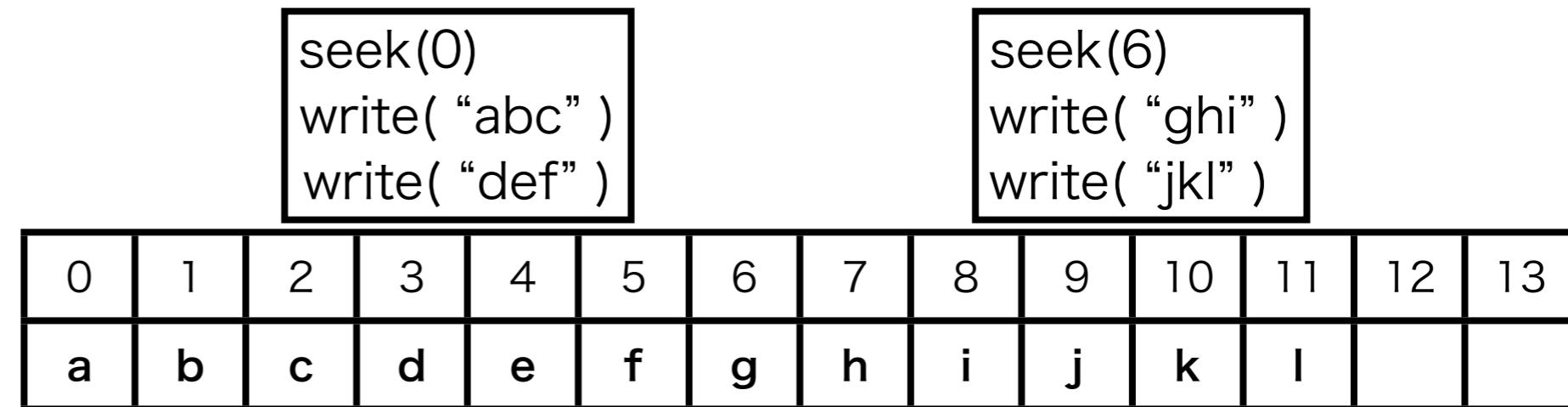


# アクセスする場所の指定の仕方（3種類）

1. FP を使わない  
で、IO 時に常に場所を指定する

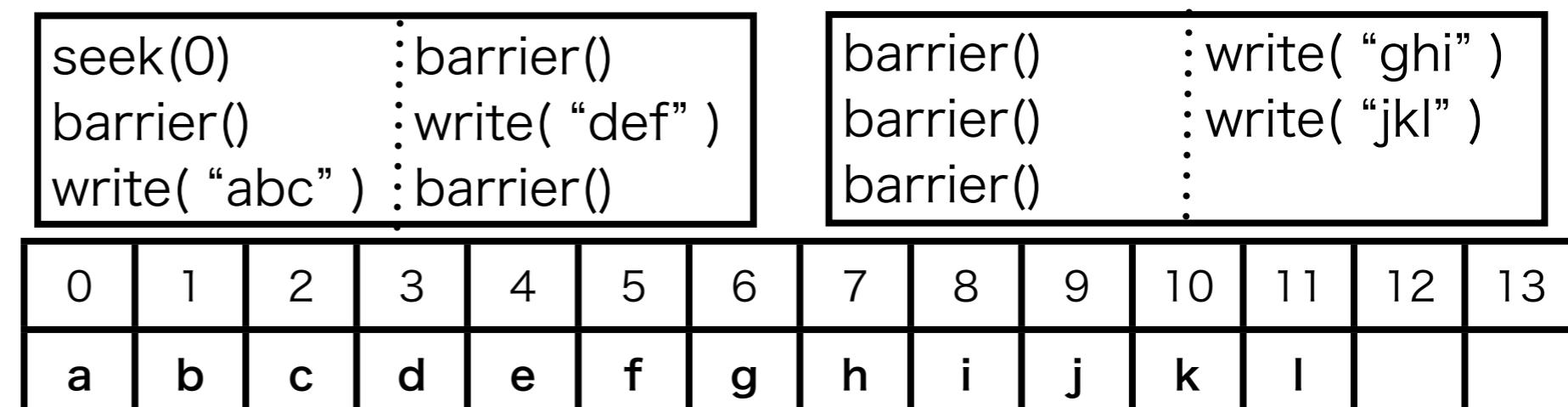


2. プロセス毎に  
FP が存在する



3. ファイルにひとつだけ FP が存在する

⇒ 遅い！！



# 2種類の File Pointer

C: MPI\_File\_seek( MPI\_File fh, MPI\_Offset off, int whence )  
MPI\_File\_seek\_shared( MPI\_File fh, MPI\_Offset off, int whence )  
MPI\_File\_get\_position( MPI\_File fh, MPI\_Offset \*off )  
MPI\_File\_get\_position\_shared (MPI\_File \*fh, MPI\_Offset \*off )

F: MPI\_FILE\_SEEK( fh, offset, whence, ierr )  
MPI\_FILE\_SEEK\_SHARED( fh, off, whence, ierr )  
MPI\_FILE\_GET\_POSITION( fh, off, ierr )  
MPI\_FILE\_GET\_POSITION\_SHARED( fh, off, ierr )

- MPI\_FILE\_SEEK
  - プロセス固有の FP を設定する
- MPI\_FILE\_SEEK\_SHARED
  - 共有 FP を設定する
- MPI\_FILE\_GET\_POSITION
  - プロセス固有の FP の値を得る
- MPI\_FILE\_GET\_POSITION\_SHARED
  - 共有 FP の値を得る

# Write\_at\_all と Read\_at\_all

```
C: MPI_File_write_at_all( MPI_File fh, MPI_Offset off, void *data,  
    int count, MPI_Datatype type, MPI_Status status )  
    MPI_File_read_at_all( MPI_File fh, MPI_Offset off, void *data,  
    int count, MPI_Datatype type, MPI_Status status )  
F: MPI_FILE_WRITE_AT_ALL( fh, offset, data, count, type,  
    status, ierr )  
    MPI_FILE_READ_AT_ALL( fh, offset, data, count, type,  
    status, ierr )
```

- 指定された Offset から、count 個数の type で  
指定されたデータ型の data を書込む／読込む。
- これらの関数では file pointer を使わない。

# ファイルの操作

C: MPI\_File\_delete( char \*filename, MPI\_Info info )  
MPI\_File\_preallocate( MPI\_File fh, MPI\_Offset size )  
MPI\_File\_get\_size( MPI\_File fh, MPI\_Offset \*size )  
F: MPI\_FILE\_DELETE( filename, info, ierr )  
MPI\_FILE\_PREALLOCATE( fh, size, ierr )  
MPI\_FILE\_GET\_SIZE( fh, size, ierr )

- MPI\_FILE\_DELETE
  - ファイルの削除
- MPI\_FILE\_PREALLOCATE
  - ファイルサイズを実際に書込む前に確保する
- MPI\_FILE\_GET\_SIZE
  - ファイルの大きさを得る

# MPI-IO のまとめ

- 3種類のアクセスする場所の指定方法
  - 陽に指定する MPI\_File\_\*\*\*\_at()
  - プロセス固有の File Pointer
    - File View の指定が可能
  - ファイルハンドルで共有される File Pointer
    - 遅いのでできるだけ使わないように
- Collective IO
  - Derived Datatype と Collective IO を組み合  
わせて使うと速くなることが多い

# 時間の計測

C: double MPI\_Wtime( void )

F: DOUBLE PRECISION MPI\_WTIME()

- 秒単位の時刻を返す

```
double dt, ts, te;
```

```
MPI_Barrier( MPI_COMM_WORLD );
```

```
ts = MPI_Wtime();
```

```
...
```

```
MPI_Barrier( MPI_COMM_WORLD );
```

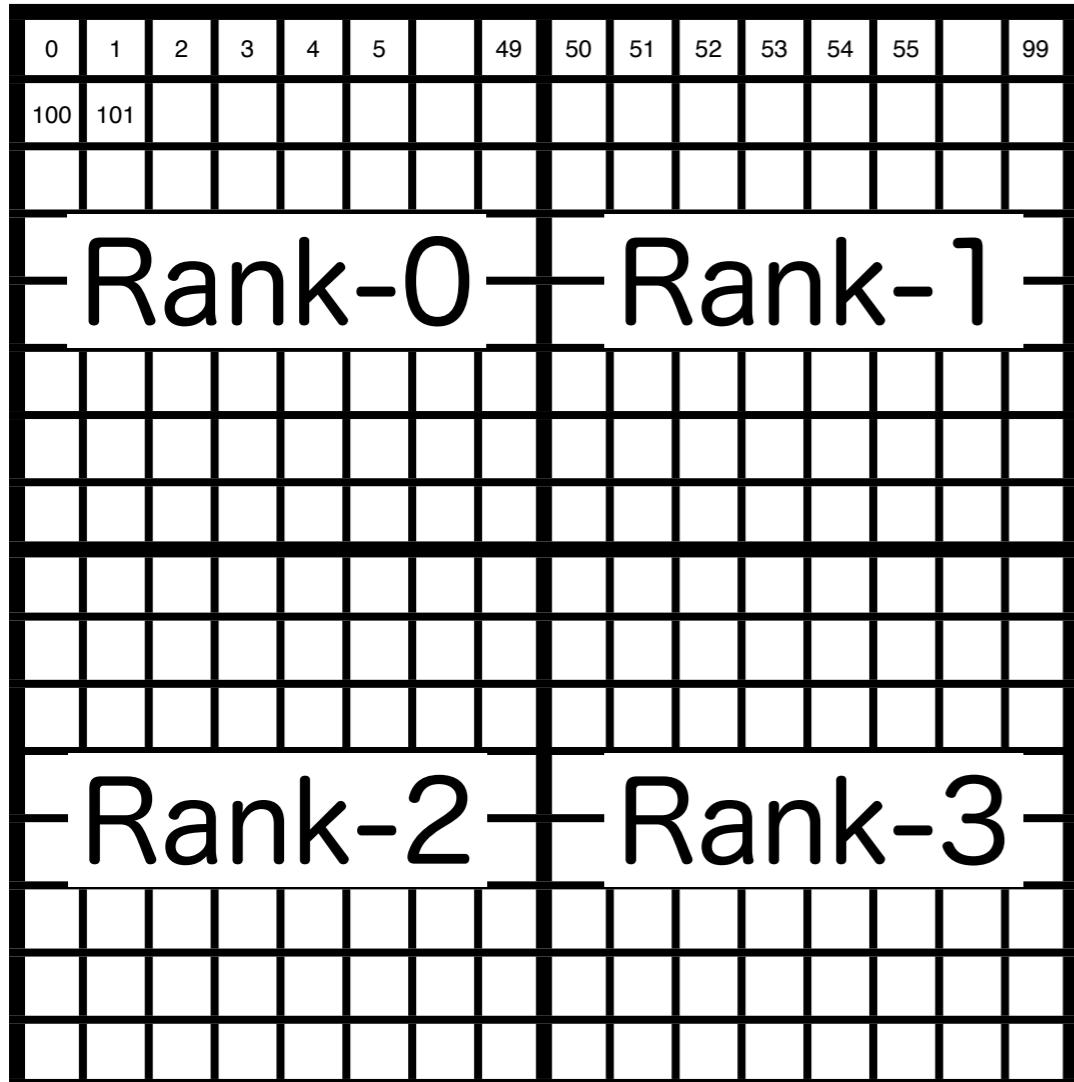
```
te = MPI_Wtime();
```

```
dt = te - ts;
```

# 質問？

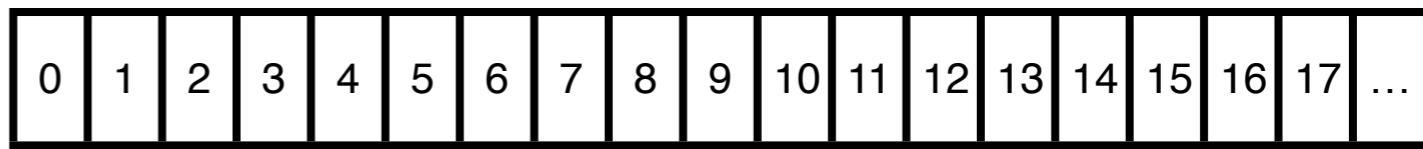
# 演習課題

- 4プロセス（ランク）のそれぞれが、下図に示す部分配列を持っていた時、MPI-IO を用いて出力するプログラムを作成せよ。



配列全体の大きさは  
100×100とする

ファイルの並びは以下  
の通り



# 演習課題

- 以下のMPI関数を用いること
  - MPI\_Type\_create\_subarray
  - MPI\_File\_set\_view
  - MPI\_File\_write\_at\_all
- 部分配列の要素の値は以下のようにする
  - Rank-0 11
  - Rank-1 13
  - Rank-2 15
  - Rank-3 17

# 余力のある人は

- 先ほど作ったファイルを、16プロセスで読込む  
プログラムを作れ。

# プログラム例

```
MPI_Init( &argc, &argv );
MPI_Comm_rank( MPI_COMM_WORLD, &rank );

nsizes[0] = LEN*2;
nsizes[1] = LEN*2;
nssizes[0] = LEN;
nssizes[1] = LEN;

switch( rank ) {
case 0:
    val = 11.0;
    starts[0] = 0;
    starts[1] = 0;
    break;
case 1:
    val = 13.0;
    starts[0] = 0;
    starts[1] = LEN;
    break;
case 2:
    val = 15.0;
    starts[0] = LEN;
    starts[1] = 0;
    break;
case 3:
    val = 17.0;
    starts[0] = LEN;

    starts[1] = LEN;
    break;
}

for( i=0; i<LEN; i++ ) {
    for( j=0; j<LEN; j++ ) {
        a[i][j] = val;
    }
}

MPI_Type_create_subarray( 2, nsizes, nssizes, starts,
                         MPI_ORDER_C, MPI_DOUBLE, &type );
MPI_Type_commit( &type );
MPI_Type_size( type, &tsz );
MPI_File_open( MPI_COMM_WORLD,
               "example.dat",
               MPI_MODE_CREATE|MPI_MODE_RDWR,
               MPI_INFO_NULL,
               &fh );
MPI_File_set_view( fh, 0, MPI_DOUBLE, type, "native",
                   MPI_INFO_NULL );
MPI_File_write_at_all( fh, 0, a, LEN*LEN, MPI_DOUBLE,
                      &status );
MPI_File_close( &fh );
```