Fig. 00.

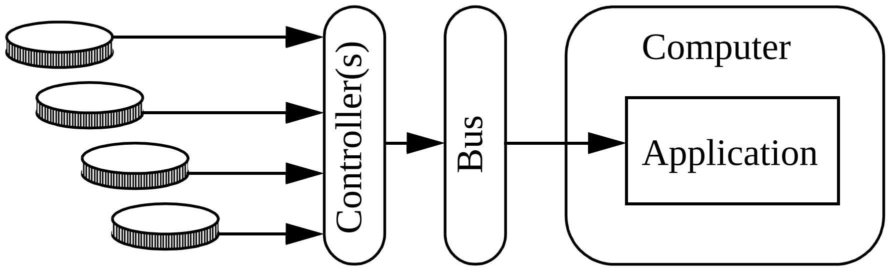
Parallel I/O for SwissTx

Emin Gabrielyan
Prof. Roger D. Hersch
Peripheral Systems Laboratory
Ecole Polytechnique Fédérale de Lausanne
Switzerland

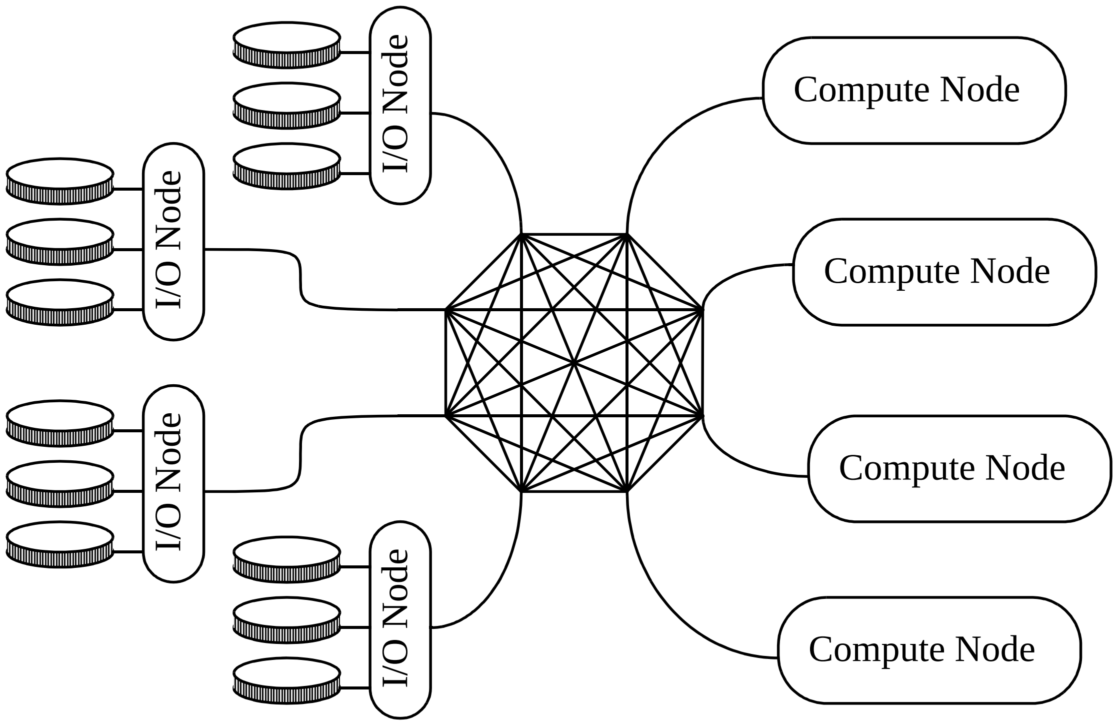
Fig. 01.

Requirements for Parallel I/O

- Scalable Throughput



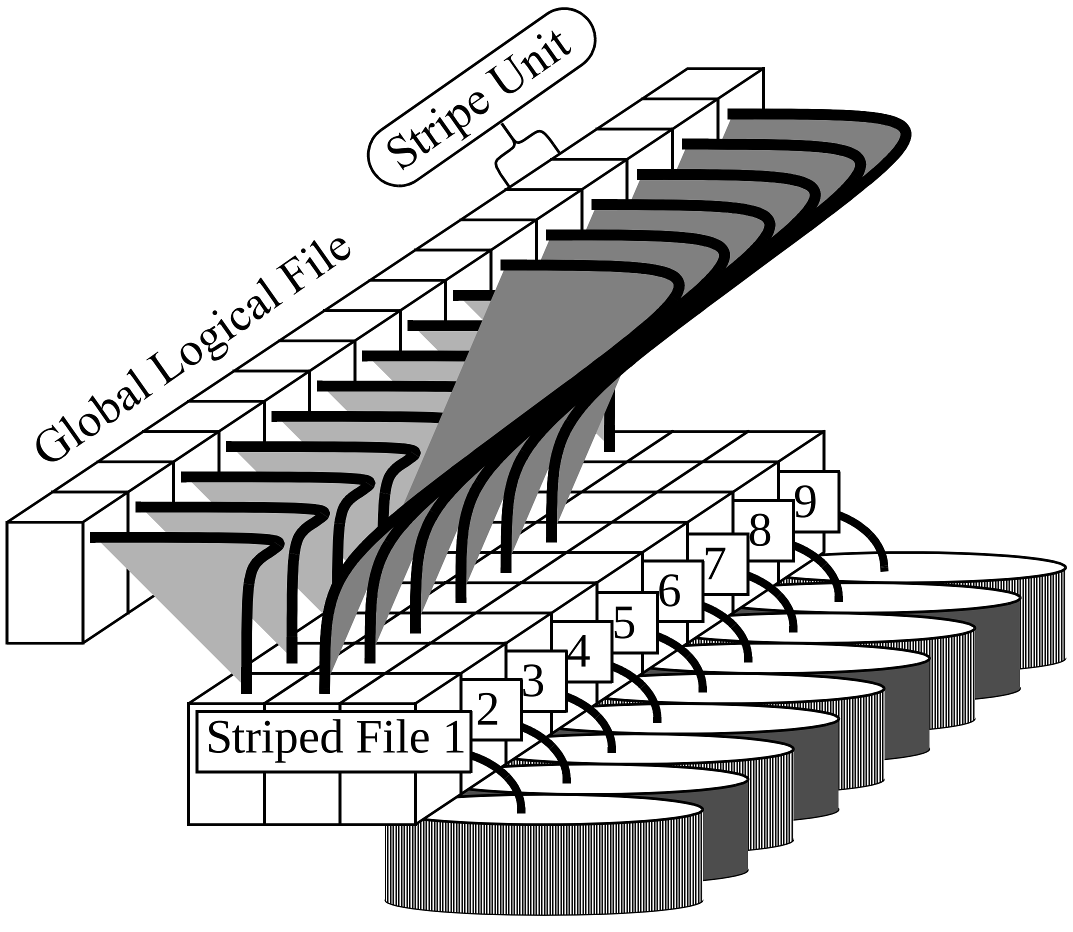
- Concurent Access



The system should be scalable. System should be adapted to concurent access

Fig. 02.

Striped Files Parallel Access Solution



The basic idea of our implementation is cyclical distribution of logical file accross the set of striped files. Access to single part of logical file require number of accesses to set of striped files, here is the space to paralelise access.

Network Transmission Optimisation

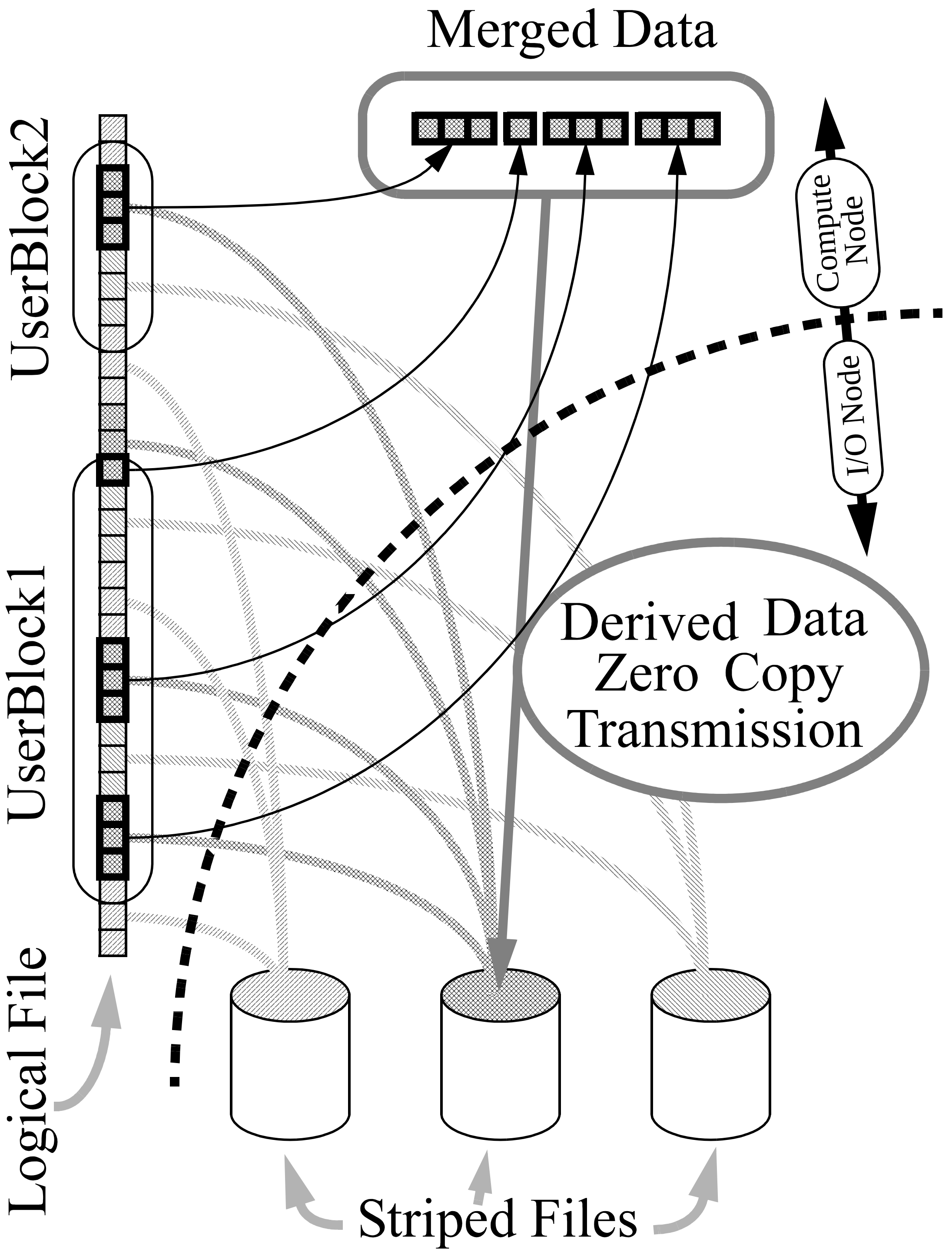
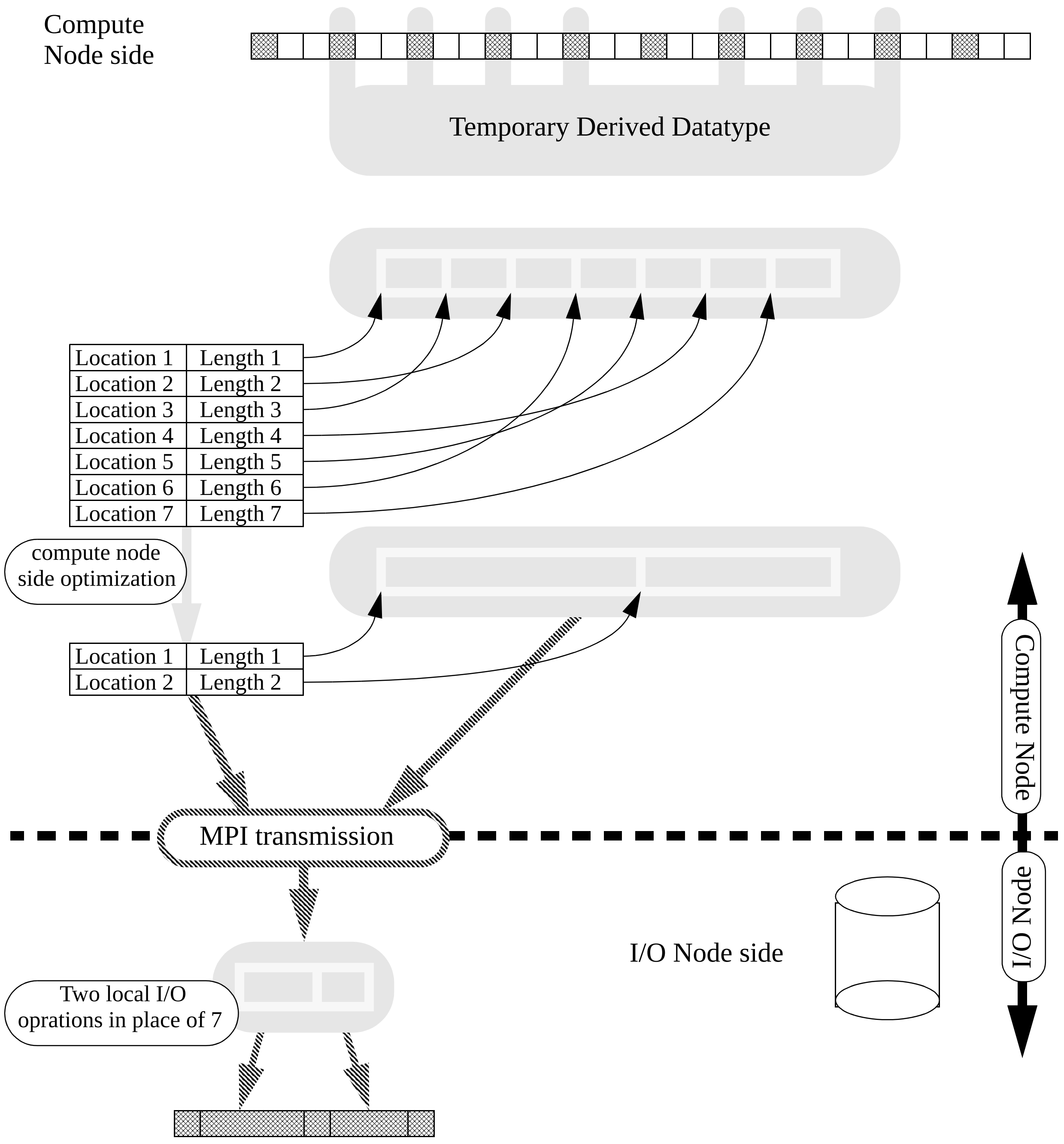


Fig. 04.

Disk Access Optimisation



The compute node before accessing I/O node try to reorgonise data so as to have minimum number of contiguous pieces

Fig. 05. Disk Access Optiisation of Collective Operations

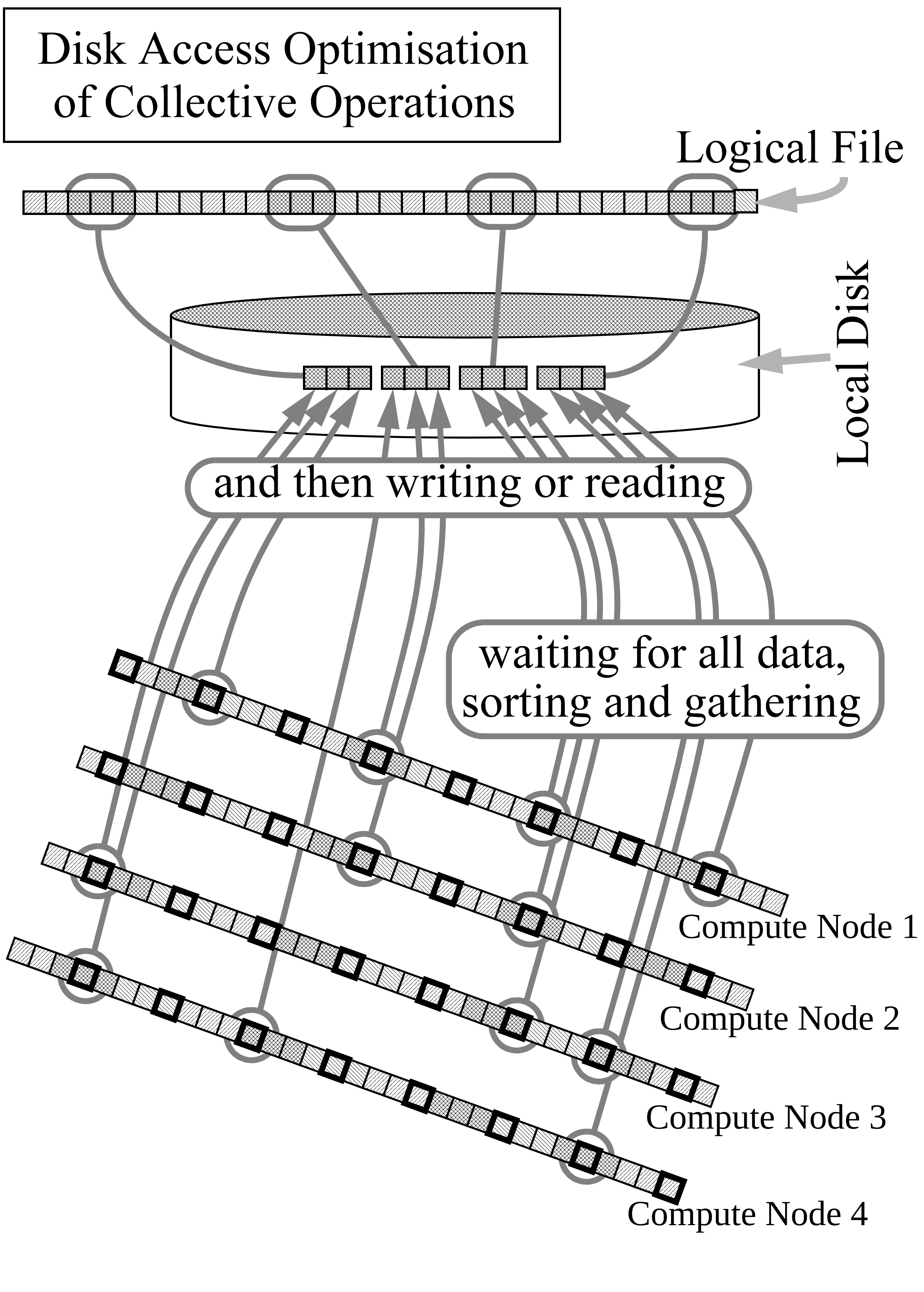

Disk Access Optimisation of Collective Operations
Logical File
Local Disk
and then writing or reading
waiting for all data, sorting and gathering
Compute Node 1
Compute Node 2
Compute Node 3
Compute Node 4

Fig. 06. Encapsulation over MPI

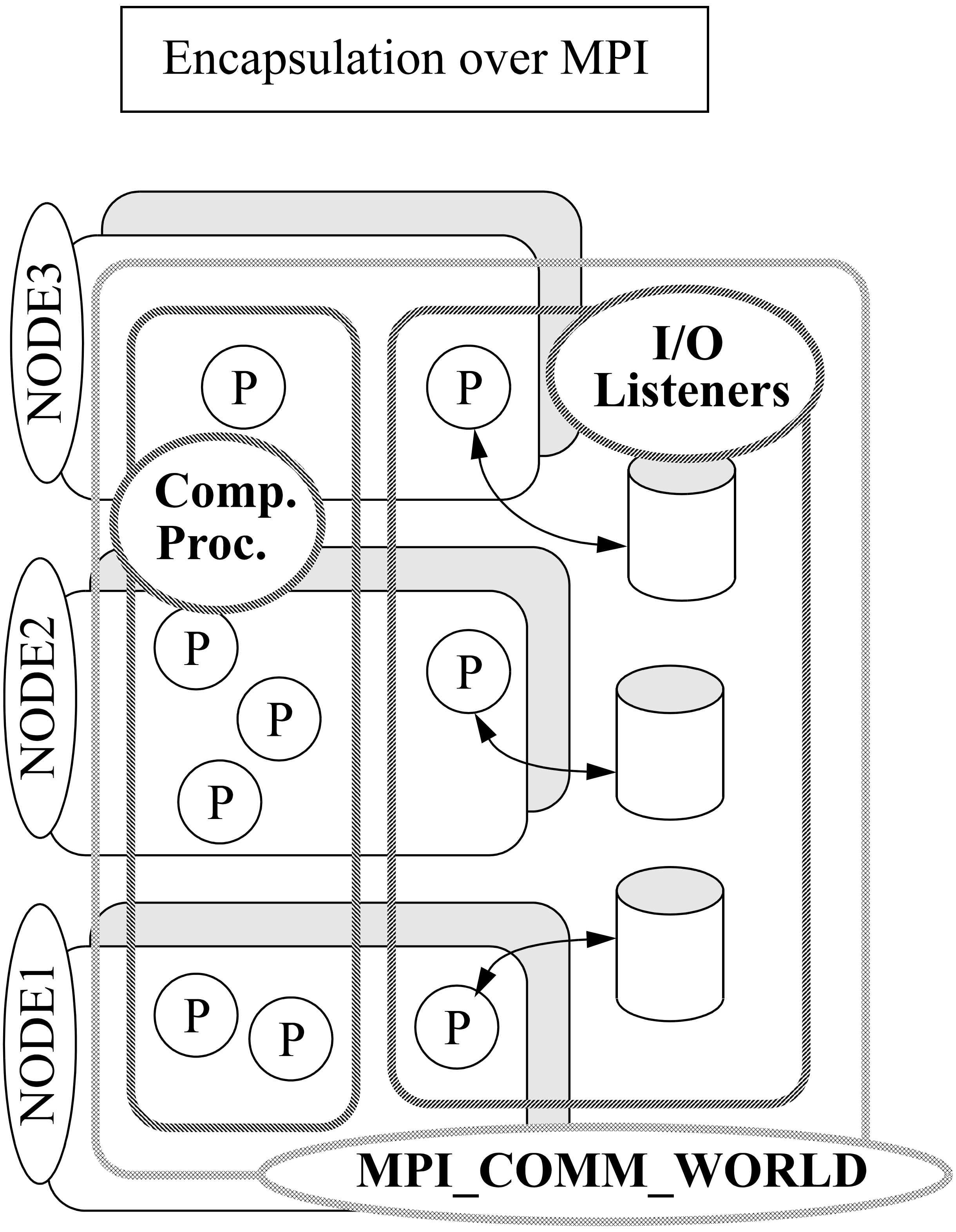
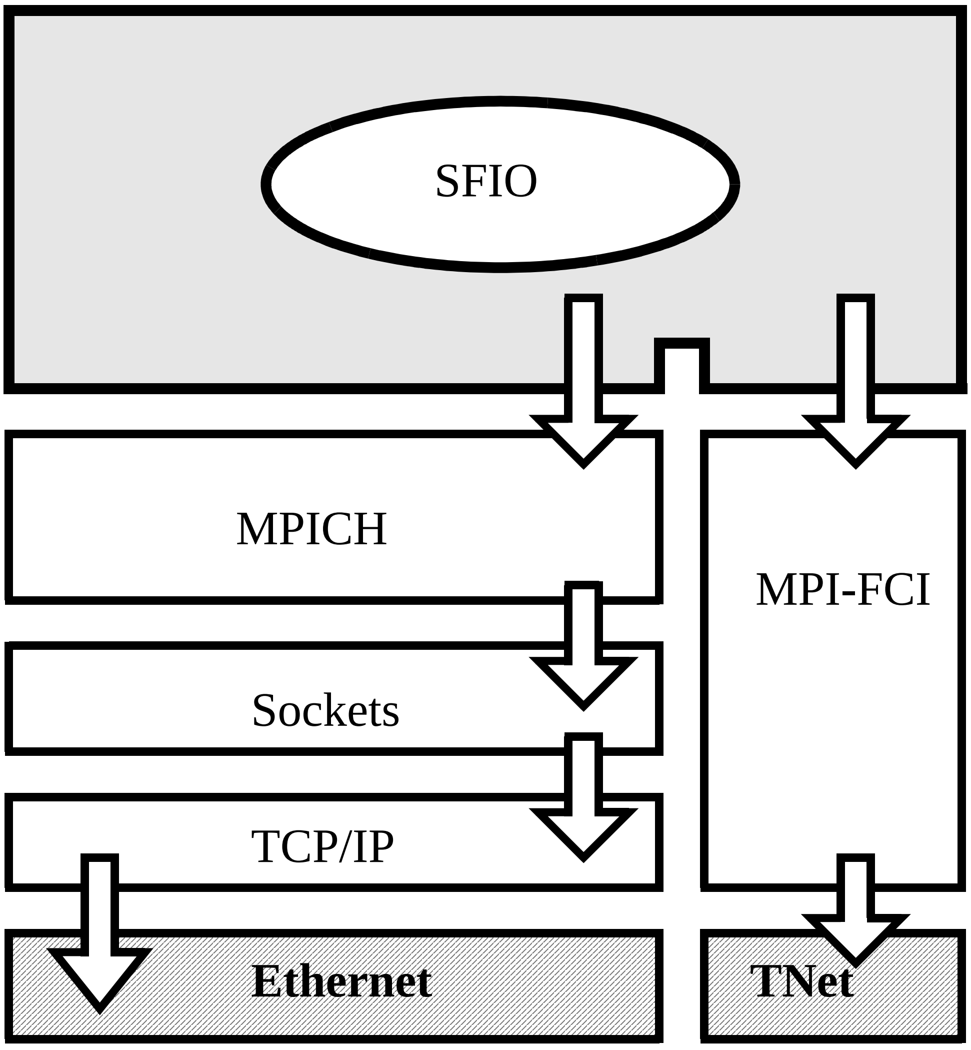


Fig. 07A. Functionality.



SFIO is implemented and tested with Digital Unix (MPICH, FCI) and Intel Windows NT (MPICH).

Fig. 07B. Functionality.

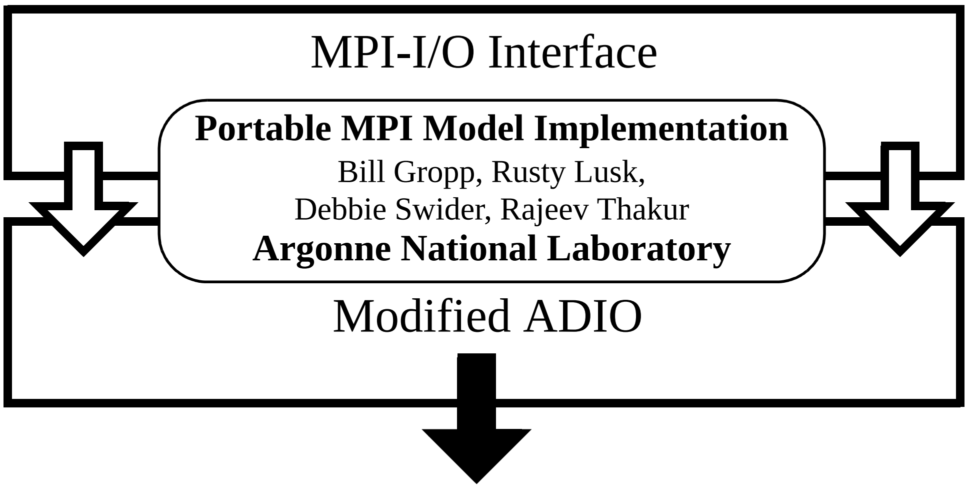

MPI-I/O Interface
Portable MPI Model Implementation
Bill Gropp, Rusty Lusk,
Debbie Swider, Rajeev Thakur
Argonne National Laboratory
Modified ADIO

Fig. 08.

Interface

● SFIO

```
MFILE* mopen(char *s, int unitsz); // “t0-p1,/tmp/a;t0-p2,/tmp/a”
void mclose(MFILE *f);
void mchsize(MFILE *f, long size);
void mdelete(char *s);
void mcreate(char *s);
void mread(MFILE *f, long offset, char *buf, unsigned count);
void mwrite(MFILE *f, long offset, char *buf, unsigned count);
void mreadb(MFILE *f, unsigned bcount,
        long Offset[], char *Buf[], unsigned Count[]);
void mwriteb(MFILE *f, unsigned bcount,
        long Offset[], char *Buf[], unsigned Count[]);
```

● MPI-II I/O

MPI_File_open	MPI_File_write_all
MPI_File_set_view	MPI_File_read_all
MPI_File_write	MPI_File_write_at_all
MPI_File_read	MPI_File_read_at_all
MPI_File_write_at	MPI_File_close
MPI_File_read_at	MPI_File_delete

The native interface is SFIO, but we work to provide also MPI-II I/O interface.

Fig. 09. Optimisation on WinNT (Fast Ethernet Switch)

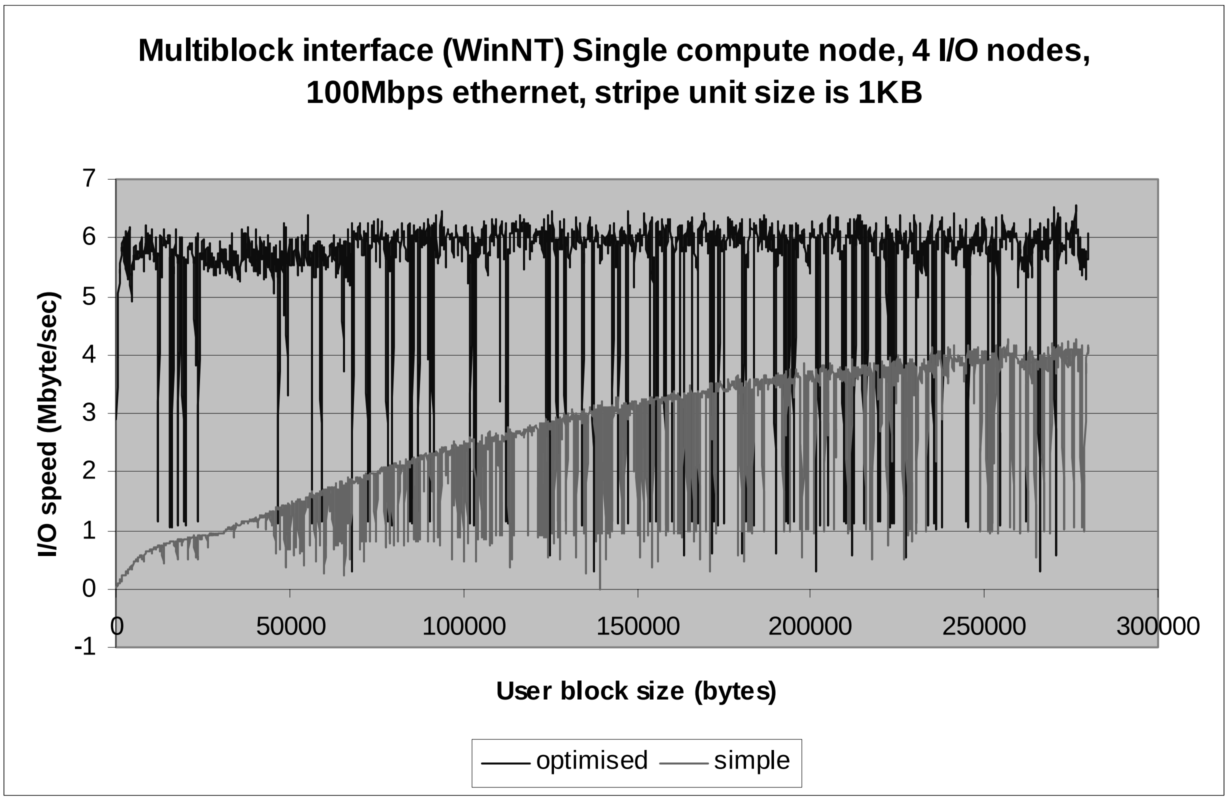


Fig. 10. Optimisation on Swiss-Tx (Fast Communication Interface)

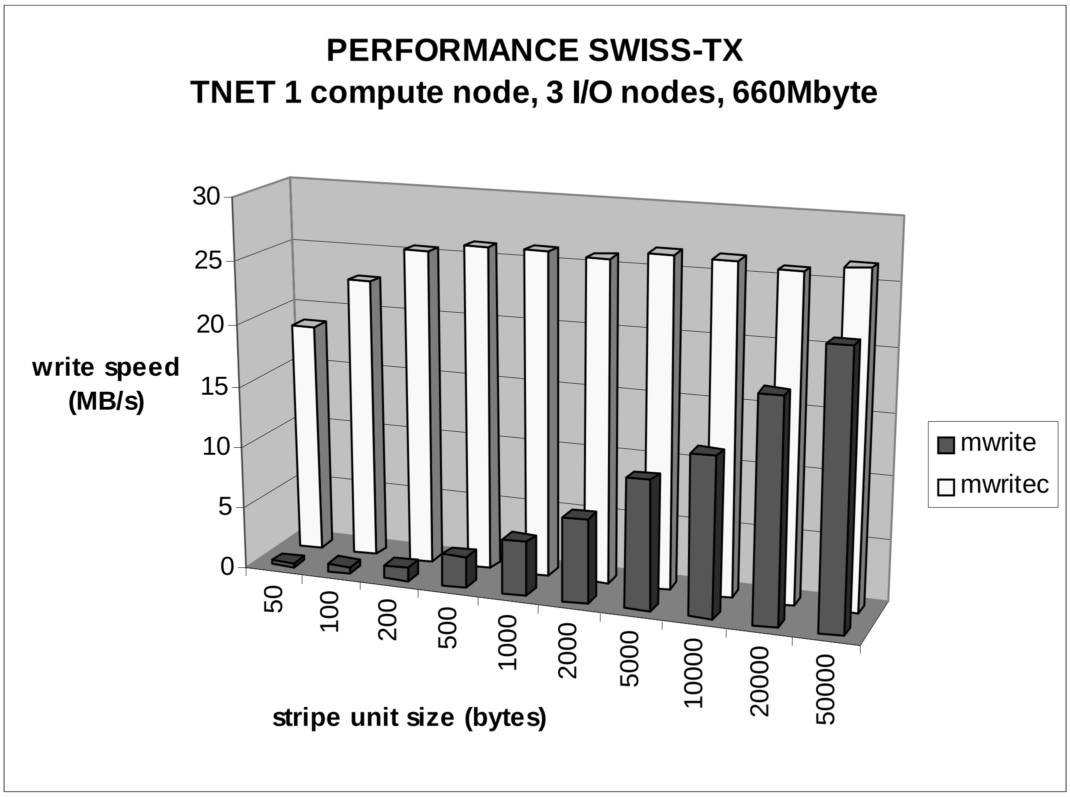
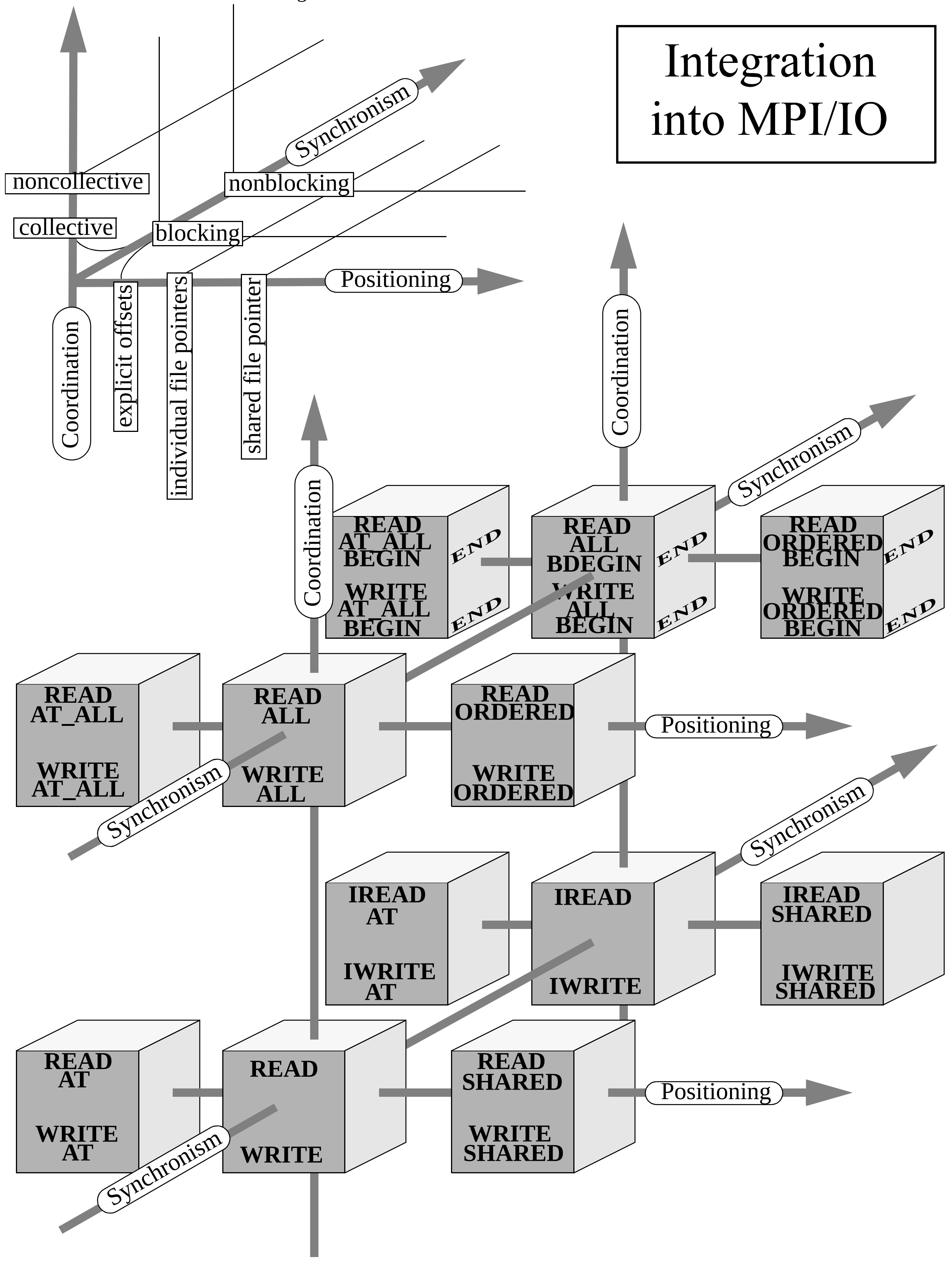


Fig. 11.


Integration into MPI/IO
noncollective
collective
nonblocking
blocking
Synchronism
Coordination
explicit offsets
individual file pointers
shared file pointer
Positioning
READ AT_ALL BEGIN
WRITE AT_ALL BEGIN
END
END
READ ALL BDEGIN
WRITE ALL BEGIN
READ ORDERED BEGIN
WRITE ORDERED BEGIN
READ AT_ALL
WRITE AT_ALL
READ ALL
WRITE ALL
READ ORDERED
WRITE ORDERED
IREAD AT
IWRITE AT
IREAD
IWRITE
IREAD SHARED
IWRITE SHARED
READ AT
WRITE AT
READ
WRITE
READ SHARED
WRITE SHARED