MPI_Init(&argc, &argv).

int MPI_Comm_rank(MPI_Comm comm, int *rank);

int MPI_Comm_size(MPI_Comm comm, int *size);

int MPI_Send(void *buf, int count, MPI_Datatype dtype, int dest, int tag, MPI_Comm comm);

int MPI_Recv(void *buf, int count, MPI_Datatype dtype, int source, int tag, MPI_Comm comm, MPI_Status *status);

int MPI_Isend(void *buf, int count, MPI_Datatype dtype, int dest, int tag, MPI_Comm comm, MPI_Request *request);

int MPI_Irecv(void *buf, int count, MPI_Datatype dtype, int source, int tag, MPI_Comm comm, MPI_Request *request);

int MPI_Wait(MPI_Request *request, MPI_Status *status);

int MPI_Test(MPI_Request *request, int *flag, MPI_Status *status);

Grupne Operacije

int MPI_Barrier (MPI_Comm comm)

int MPI_Reduce (void* send_buffer, void* recv_buffer, int count, MPI_Datatype datatype, MPI_Op operation, int rank, MPI_Comm comm)

int MPI_Scan(void* send_buffer, void* recv_buffer, int count, MPI_Datatype datatype, MPI_Op operation, MPI_Comm comm)

int MPI_Bcast (void* buffer, int count, MPI_Datatype datatype, int rank, MPI_Comm comm)

int MPI_Scatter (void* send_buffer, int send_count, MPI_datatype send_type, void* recv_buffer, int recv_count, MPI_Datatype recv_type, int rank, MPI_Comm comm)

int MPI_Gather (void* send_buffer, int send_count, MPI_datatype send_type, void* recv_buffer, int recv_count, MPI_Datatype recv_type, int rank, MPI_Comm comm)

Tipovi

int MPI_Type_struct(int count, int *array_of_blocklengths, MPI_Aint *array_of_displacements, MPI_Datatype *array_of_types, MPI_Datatype *newtype)

MPI_Address(void *location, MPI_Aint * adress)

int MPI_Type_contiguous(int count, MPI_Datatype oldtype, MPI_Datatype *newtype)

int MPI_Type_vector(int count, int blocklength, int stride, MPI_Datatype oldtype, MPI_Datatype *newtype)

int MPI_Type_indexed(int count, int *array_of_blocklengths, int *array_of_displacements, MPI_Datatype oldtype, MPI_Datatype *newtype)

int MPI_Type_create_subarray(int ndims, int *sizes, int * subsizes, int *offsets, int order, MPI_Datatype oldtype, MPI_Datatype *newtype)

int MPI_Type_commit (MPI_datatype *datatype)

int MPI_Type_create_resized(MPI_Datatype oldtype, MPI_Aint lb, MPI_Aint extent, MPI_Datatype *newtype)

Grupe

int MPI_Comm_group(MPI_Comm comm, MPI_Group *group)

int MPI_Group_rank(MPI_Group group, int *rank)

int MPI_Group_size(MPI_Group group, int *size)

int MPI_Group_excl(MPI_Group group, int count, int *nonmembers, MPI_Group *new_group)

int MPI_Group_incl(MPI_Group old_group, int count, int *members, MPI_Group *new_group)

int MPI_Group_intersection(MPI_Group group1, MPI_Group group2, MPI_Group *newgroup)

int MPI_Group_union(MPI_Group group1, MPI_Group group2, MPI_Group *newgroup)

int MPI_Group_difference(MPI_Group group1, MPI_Group group2, MPI_Group *newgroup)

Komunikatori

int MPI_Comm_create(MPI_Comm old_comm, MPI_Group, MPI_Comm * new_comm)

int MPI_Comm_split(MPI_Comm, int color, int key, MPI_Comm *new_comm)

Topologije

int MPI_Cart_create(MPI_Comm old_comm, int ndims, int *dim_size, int *periods, int reorder, MPI_Comm *new_comm)

int MPI_Cart_coords(MPI_Comm comm, int rank, int maxdims, int *coords)

int MPI_Cart_rank (MPI_Comm comm, init *coords, int *rank)

int MPI_Cart_shift(MPI_Comm comm, int direction, int disp, int *rank_source, int *rank_dest)

int MPI_Sendrecv(const void *sendbuf, int sendcount, MPI_Datatype sendtype, int dest, int sendtag, void *recvbuf, int recvcount, MPI_Datatype recvtype, int source, int recvtag, MPI_Comm comm, MPI_Status *status)

int MPI_Sendrecv_replace(void *buf, int count, MPI_Datatype datatype, int dest, int sendtag, int source, int recvtag, MPI_Comm comm, MPI_Status * status)

Fajlovi

int MPI_File_open(MPI_Comm comm, const char *filename, int amode, MPI_Info info, MPI_File *fh)

MPI_MODE_RDONLY – Read only
MPI_MODE_RDWR – Read and Write
MPI_MODE_WRONLY - Write only
MPI _MODE_CREATE - Create file if it doesn’t exist

int MPI_File_seek(MPI_File fh, MPI_Offset offset, int whence)

Whence
MPI_SEEK_SET - Pointer se postavlja na vrednost pomeraja (od početka fajla)
MPI_SEEK_CUR - Pointer se postavlja na trenutnu poziciju pokazivača + pomeraj
MPI_SEEK_END - Pointer se postavlja na kraj fajla + pomeraj

int MPI_File_read(MPI_File fh, void *buf, int count, MPI_Datatype datatype, MPI_Status *status)

int MPI_File_write(MPI_File fh, const void *buf, int count, MPI_Datatype datatype, MPI_Status *status)

int MPI_File_close(MPI_File *fh)

int MPI_File_read_at(MPI_File fh, MPI_Offset offset, void *buf, int count, MPI_Datatype datatype, MPI_Status *status)

int MPI_File_write_at(MPI_File fh, MPI_Offset offset, const void *buf, int count, MPI_Datatype datatype, MPI_Status *status)

Nekontinualni pristup

int MPI_File_set_view(MPI_File fh, MPI_Offset disp, MPI_Datatype etype, MPI_Datatype filetype, const char * datarep, MPI_Info info)

Grupne I/O operacije

- Svi u isto vreme pristupaju
int MPI_File_read_all(MPI_File fh, void *buf, int count, MPI_Datatype datatype, MPI_Status *status)

int MPI_File_write_all(MPI_File fh, const void *buf, int count, MPI_Datatype datatype, MPI_Status *status)

int MPI_Type_create_darray(int size, int rank, int ndims, int gsizes[], int distribs[], int dargs[], int psizes[], int order, MPI_Datatype oldtype, MPI_Datatype *newtype)

• Size - Broj procesa među kojima je niz distribuiran
• Ndims - Broj dimenzija polja i potpolja (N)
• gsizes - Broj elemenata starog tipa (oldtype) u svakoj dimenziji polja (niz pozitivnih celih brojeva)
• distribs - Način distribucije- MPI_DISTRIBUTE_BLOCK
• dargs - Parametar distribucije za svaku dimenziju- MPI_DISTRIBUTE_DFLT_DARG
• psizes - Broj procesa u svakoj dimenziji među kojima je polje distribuirano. Uzima se da grid procesa ima isti broj dimenzija kao polje i potpolje. Ako niz po nekoj dimenziji nije distribuiran, broj procesa u toj dimenziji je 1!
• Order - Način predstavljanja polja u memoriji, MPI_ORDER_C

int MPI_Type_create_subarray(int ndims, int *sizes, int *subsizes, int *offsets, int order, MPI_Datatype oldtype, MPI_Datatype *newtype)

• ndims - broj dimenzija polja (N)(pozitivan broj)
• sizes - broj elemenata starog tipa (oldtype) u svakoj dimenziji polja (niz pozitivnih celih brojeva)
• subsizes - broj elemenata starog tipa (oldtype) u svakoj dimenziji potpolja (niz pozitivnih celih brojeva)
• offsets - početne koordinate podpolja u svakoj dimenziji(niz nenegativnih brojeva)
• order - način predstavljanja polja u memoriji, ili MPI_ORDER_C ili MPI_ORDER_FORTRAN

Neblokirajuce I/O operacije

int MPI_File_iread(MPI_File fh, void *buf, int count, MPI_Datatype datatype, MPI_Request* request)

int MPI_File_iwrite(MPI_File fh, ROMIO_CONST void *buf, int count, MPI_Datatype datatype, MPI_Request* request)

int MPI_File_iwrite_at(MPI_File fh, MPI_Offset offset, const void *buf, int count,MPI_Datatype datatype, MPI_Request* request);

int MPI_File_iread_at(MPI_File fh, MPI_Offset offset, void *buf, int count, MPI_Datatype datatype, MPI_Request* request)

Grupne I neblokirajuce operacije

Kako bi se koristile grupne neblokirajuće I/O operacije, korisnik mora definisati početak i kraj operacije:

int MPI_File_write_all_begin(MPI_File fh, const void *buf, int count, MPI_Datatype datatype);

int MPI_File_write_aIl_end(MPI_File fh, const void *buf, MPI_Status *status);

Ograničenje– U jednom trenutku može biti aktivna samo jedna split collective I/O operacija nad jednim fajlom

Deljeni pokazivac

Funkcije za čitanje/upis sa trenutne pozicije deljenog pokazivača:

int MPI_File_write_shared(MPI_File fh, const void *buf, int count, MPI_Datatype datatype, MPI_Status *status);

int MPI_File_read_shared(MPI_File fh, void *buf, int count, MPI_Datatype datatype, MPI_Status * status);

int MPI_File_seek_shared(MPI_File fh, MPI_Offset offset, int whence);

Nakon svakog poziva ovih funkcija, pozicija pokazivača se ažurira za količinu upisanih/pročitanih podataka
Naredni poziv funkcije bilo kog procesa iz grupe čita/upisuje podatke na novu poziciju pokazivača
NEMA REDOSLEDA PREDEFINISANOG!

Deljeni pokazivac I neblokirajuce operacije

int MPI_File_iwrite_shared(MPI_File fh, ROMIO_CONST void *buf, int count, MPI_Datatype datatype, MPI_Request *request)

int MPI_File_iread_shared(MPI_File fh, void *buf, int count, MPI_Datatype datatype, MPI_Request *request)

Deljeni pokazivac I grupne operacije

int MPI_File_read_ordered(MPI_File fh, void *buf, int count, MPI_Datatype datatype, MPI_Status *status)

int MPI_File_write_ordered(MPI_File fh, ROMIO_CONST void *buf, int count, MPI_Datatype datatype, MPI_Status *status)

Upis i čitanje se u ovom slučaju vrše redom, po ranku procesa!

Sinhronizacija

MPI_File_set_atomicity(MPI_File fh, int flag)
Poziv ove funkcije pre upisa garantuje da se čitanje može obaviti odmah nakon upisa.

Obezbediti da nijedna sekvenca upisa nije konkurentna sa sekvencom upisa/čitanja drugog procesa
Sekvencom se smatra grupa operacija između dva poziva MPI_File_open, MPI_File_close ili MPI_File_sync funkcija

MPI_File_sync(MPI_File fh)
Ovo je grupna operacija, mora se pozvati iz oba procesa. Ovakav pristup nije moguć kod grupnih operacija!