



Επιστημονικοί Υπολογισμοί - Μέρος III: Παράλληλοι Υπολογισμοί

Χαρμανδάρης Βαγγέλης, Τμήμα Εφαρμοσμένων Μαθηματικών
Πανεπιστήμιο Κρήτης, Εαρινό Εξάμηνο 2013/14

Κεφάλαιο 5: (B) Message Passing Interface (MPI)



Message Passing Interface (MPI)

Βασική ιδέα: Ο χρήστης γράφει τον κώδικα (π.χ. σε C ή Fortran) και μετά χρησιμοποιεί τις εντολές της κατάλληλης βιβλιοθήκης για τον παραλληλισμό των διεργασιών.

Θα δούμε πιο αναλυτικά:

- **Τι είναι το MPI; Εισαγωγικά.**
- **Ένα απλό πρόγραμμα σε MPI.**
- **Υποπρογράμματα Διαχείρισης του MPI Παράλληλου Περιβάλλοντος.**
- **Υποπρογράμματα Επικοινωνίας των Διεργασιών.**
- **Τοπολογία**



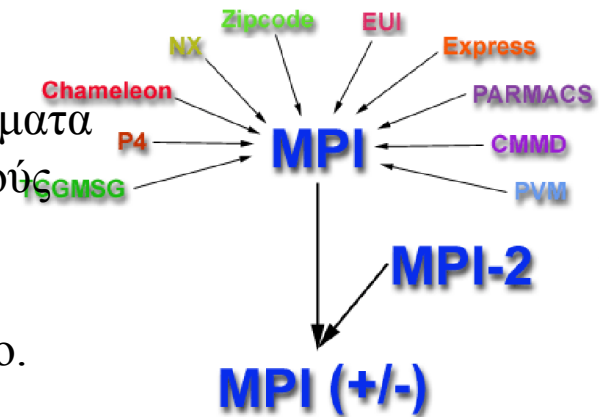
Τι Είναι το MPI;

- Το MPI είναι μια βιβλιοθήκη υποπρογραμμάτων ανταλλαγής μηνυμάτων και μεταφοράς δεδομένων.
- Χαρακτηριστικά του MPI: Αποτελεσματικότητα, Φορητότητα-Μεταβιβασιμότητα, Ευελιξία.
- Ανεπτυγμένο για πληθώρα (σχεδόν όλα) υπολογιστικών συστημάτων διαφορετικής αρχιτεκτονικής.
- Σχεδιασμένο το για C/C++ όσο και Fortran/ Fortran 90.

- Ιστορία και Εξέλιξη:

- 1980 – αρχές 1990: Ανομοιογενές λογισμικό. Προβλήματα μεταφοράς προγραμμάτων. Ανάγκη δημιουργίας διεθνούς προτύπου.

- 1992 – 1994: Καθιέρωση του MPI ως διεθνές πρότυπο.





Τι Είναι το MPI;

Λόγοι χρησιμοποίησης του MPI:

- **Διεθνές Πρότυπο:** Το MPI είναι η στάνταρτ βιβλιοθήκη ανταλλαγής μηνυμάτων. Υποστηρίζεται από όλα τα υπολογιστικά συστήματα HPC (High Performance Computing).
- **Διαθεσιμότητα:** Υπάρχουν διάφορες διαθέσιμες υλοποιήσεις (implementations). Το λογισμικό είναι ελεύθερο.
- **Αποτελεσματικότητα:** οι διαφορετικές υλοποιήσεις εκμεταλλεύονται χαρακτηριστικά του hardware με αποτέλεσμα την καλύτερη απόδοση των προγραμμάτων.
- **Φορητότητα:** οι κώδικες μεταφέρονται εύκολα σε διαφορετικά συστήματα.
- **Λειτουργικότητα:** Στο MPI-1 υπάρχουν 128 υποπρογράμματα και στο MPI-2 152.
- Το MPI είναι ανεπτυγμένο για συστήματα κοινής και κατανεμημένης μνήμης.



Γενική Δομή ενός Προγράμματος MPI

MPI include file

Declarations, prototypes, etc.

Program Begins

.
.
.
Serial code

Initialize MPI environment

Parallel code begins

.
.
.

Do work and make message passing calls

.
.
.

Terminate MPI Environment

Parallel code ends

.
.
.
Serial code

Program Ends



Παράδειγμα: 'Hello world' πρόγραμμα

Απλό πρόγραμμα σε C

```
#include <stdio.h>
#include "mpi.h"

int main(argc,argv)
{
    int numtasks, rank,

    MPI_Init (&argc,&argv);

    MPI_Comm_size(MPI_COMM_WORLD, &numtasks);
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);

    printf ("Hello world from process %d of %d\n", rank, numtasks);

    MPI_Finalize();
}
```



Παράδειγμα: 'Hello world' πρόγραμμα

Απλό πρόγραμμα σε Fortran

```
program simple

include 'mpif.h'

integer numtasks, rank, ierr

call MPI_INIT(ierr)

call MPI_COMM_SIZE(MPI_COMM_WORLD, numtasks, ierr)
call MPI_COMM_RANK(MPI_COMM_WORLD, rank, ierr)

print *, 'Hello world from process ', rank, ' of ', numtasks

call MPI_FINALIZE(ierr)

stop
end
```



Περιγραφή Απλού Προγράμματος

- Οι εντολές

#include "mpi.h", include 'mpif.h'

απαιτούνται από **όλα τα προγράμματα / υπορουτίνες που χρησιμοποιούν MPI** καθώς είναι αυτές που παρέχουν πρόσβαση στην βιβλιοθήκη MPI.

Η διάταξη των συναρτήσεων MPI είναι:

- **C/C++:** *MPI_Xxxxx (parameter, ...)*

➤ Παράδειγμα: *MPI_Send (&buf, count, type, dest, comm)*

➤ Όλες οι συναρτήσεις C του MPI επιστρέφουν δείκτες λαθών (error code) τύπου ακεραίου.

- **Fortran:** *call MPI_XXXX (parameter, ..., ierr)*

➤ Παράδειγμα: *MPI_SEND (buf, count, type, dest, comm, ierr)*

➤ Τα υποπρογράμματα Fortran επιστρέφουν δείκτες λαθών τύπου ακεραίου ως το τελευταίο όρισμά τους.

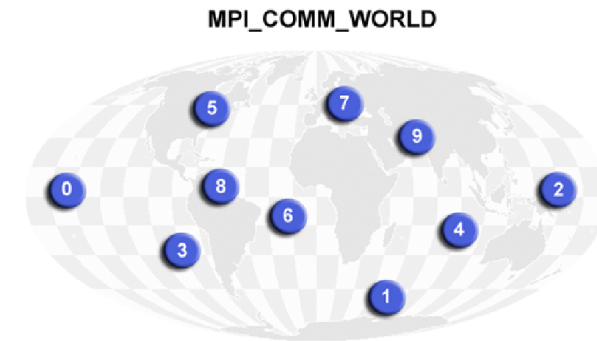
- Οι κοινές εντολές του προγράμματος (π.χ. `print *`, `'Hello world from process ', rank, 'of', numtasks`) είναι **τοπικές**, δηλαδή **εκτελούνται από κάθε επεξεργαστή**.



Περιγραφή Απλού Προγράμματος: Υποπρογράμματα Διαχείρισης

Διαχειριστής Επικοινωνίας (Δ.Ε.) - Communicator: Ένα σύνολο διεργασιών που μπορούν να στείλουν/λάβουν μηνύματα.

- **MPI_COMM_WORLD** είναι ο προκαθορισμένος Δ.Ε.



- Συνήθως ο αριθμός των διεργασιών συμπίπτει με τον αριθμό των επεξεργαστών που θέλουμε να χρησιμοποιήσουμε. Οι επεξεργαστές μπορεί να είναι οπουδήποτε.
- **Ταξινόμηση:** Μέσα στον Δ.Ε. κάθε διεργασία αριθμείται με ένα ακέραιο αριθμό (rank) ξεκινώντας από το 0. Δηλαδή P διεργασίες αριθμούνται από 0 ως $P-1$.
- Η ταξινόμηση χρησιμοποιείται από τον χρήστη στην επικοινωνία μεταξύ των διεργασιών (π.χ. `if rank=0 do something / if rank=1 do something else`).



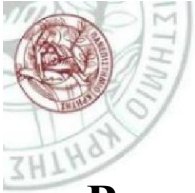
Υποπρογράμματα-Συναρτήσεις Διαχείρισης MPI

- **MPI_Init:** Υποδηλώνει την έναρξη του MPI. Πρέπει να καλείται πριν από οποιαδήποτε άλλη εντολή του MPI.
C: *MPI_Init (&argc, &argv);*
Fortran: *call MPI_INIT(ierr)*
- **MPI_Comm_size:** Προσδιορίζει τον αριθμό των διεργασιών-επεξεργαστών που σχετίζονται με τον ΔΕ.
C: *MPI_Comm_size (comm, &size);*
Fortran: *call MPI_COMM_SIZE(comm, size, ierr)*
- **MPI_Comm_rank:** Προσδιορίζει τον αριθμό της διεργασίας που καλείται μέσα στον ΔΕ.
C: *MPI_Comm_rank (comm, &rank);*
Fortran: *call MPI_COMM_RANK(comm, rank, ierr)*



Υποπρογράμματα Διαχείρισης MPI

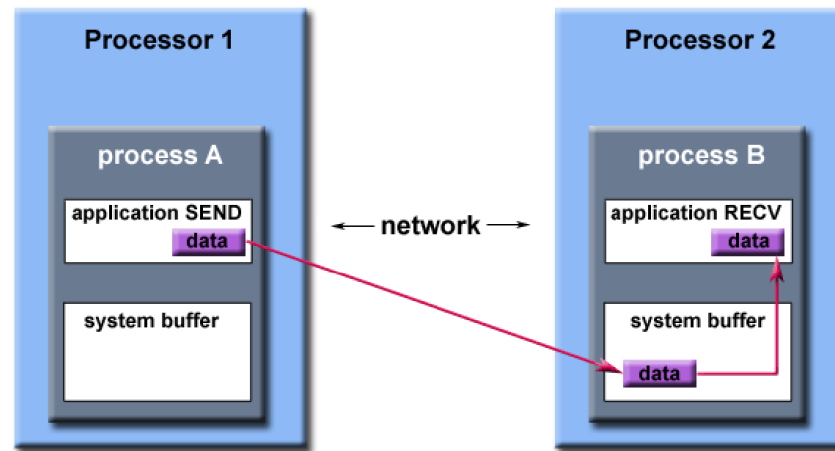
- **MPI_Abort:** Τερματίζει όλες τις διεργασίες MPI που σχετίζονται με τον Δ.Ε.
C: *MPI_Abort (comm, ierr);*
Fortran: *call MPI_ABORT(comm, ierr)*
- **MPI_Get_processor_name:** Επιστρέφει το όνομα του επεξεργαστή καθώς και το μήκος (αριθμός χαρακτήρων) του ονόματος.
C: *MPI_Get_processor_name (&name, &namelength);*
Fortran: *call MPI_GET_PROCESSOR_NAME(name, namelength)*
- **MPI_Wtime:** Επιστρέφει τον χρόνο (wall time clock), σε δευτερόλεπτα ενός επεξεργαστή. Χρησιμοποιείται για μέτρηση του χρόνου εκτέλεσης μιας διεργασίας.
C: *MPI_Wtime ();*
Fortran: *call MPI_WTIME()*
- **MPI_Finalize:** Υποδηλώνει τη λήξη χρήσης του MPI. Πρέπει να καλείται στο τέλος.
C: *MPI_Finalize (&argc, &argv);*
Fortran: *call MPI_FINALIZE(ierr)*



Υποπρογράμματα Επικοινωνίας MPI

Βασικά στοιχεία:

- Ποιος στέλνει το μήνυμα / που στάλθηκε το μήνυμα.
- Μέγεθος – Τύπος των δεδομένων που στάλθηκαν.
- Λήψη του μηνύματος – Αναγνώριση των δεδομένων.



Path of a message buffered at the receiving process

Τύποι επικοινωνίας:

- Μια-προς-μια επικοινωνία (**Point-to-Point Communication**).
- Συλλογική επικοινωνία (**Collective Communication**).



Μια-προς-μια Επικοινωνία Point-to-Point Communication

- Η αποστολή / λήψη μηνυμάτων γίνεται ανάμεσα από μόνο δύο διεργασίες. Μια διεργασία εκτελεί την αποστολή και μια την λήψη.
- Διάφοροι τύποι συναρτήσεων αποστολής/λήψης μηνυμάτων:
 - Συγχρονισμένη αποστολή (Synchronous send).
 - Περιορισμένη αποστολή / περιορισμένη λήψη (Blocking send / Blocking receive).
 - Μη-περιορισμένη αποστολή / Μη-περιορισμένη λήψη (Non-blocking send / Non-blocking receive).
 - Συνδυασμένη αποστολή/λήψη (Combined send/receive).
- Κάθε τύπος συνάρτησης αποστολής μπορεί να συνδυαστεί με οποιοδήποτε τύπο συνάρτησης λήψης.
- Υπάρχουν επίσης αρκετές συναρτήσεις-υποπρογράμματα παρακολούθησης και ελέγχου αποστολής/λήψης μηνυμάτων.



Point-to-Point Communication / Buffering

- Θεωρητικά κάθε εντολή αποστολής μπορεί να είναι συγχρονισμένη με την εντολή λήψης. Όμως αυτό είναι πολύ σπάνιο.
- Παράδειγμα:
 - Μια εντολή αποστολής εκτελείται 5 δευτερόλεπτα πριν την αντίστοιχη εντολή λήψης. Που βρίσκονται τα δεδομένα κατά τη διάρκεια της αναμονής;
 - Πολλαπλά μηνύματα φτάνουν στη ίδια διεργασία. Τι γίνεται με τα δεδομένα που βρίσκονται σε αναμονή;
- Η υλοποίηση του MPI αποφασίζει τι συμβαίνει. Συνήθως δεσμεύεται ένας Προσωρινός Χώρος Μνήμης Συστήματος (ΠΧΜΣ) (**System Buffer**).
- Προσωρινός Χώρος Μνήμης Συστήματος:
 - Διαχειρίζεται από το MPI, ο χρήστης δεν έχει πρόσβαση σε αυτόν.
 - Έχει μικρή χωρητικότητα.
 - Χρησιμοποιείται από εντολές αποστολής και λήψης.
 - Βελτιώνει την απόδοση του προγράμματος.
- Ο χώρος μνήμης που δεσμεύουν οι μεταβλητές του κώδικα ονομάζεται Προσωρινός Χώρος Μνήμης Εφαρμογής (ΠΧΜΕ) (**Application Buffer**).



Περιορισμένη/Μη-περιορισμένη Αποστολή/Λήψη

Περιορισμένη αποστολή/λήψη μηνυμάτων:

- Μια συνάρτηση περιορισμένης αποστολής θα «τερματίσει» (return) μόνο όταν ο ΠΧΜΕ μπορεί να επαναχρησιμοποιηθεί.
- Μπορεί να είναι συγχρονισμένη: υπάρχει επικοινωνία με την διεργασία που λαμβάνει για ασφαλή παραλαβή των δεδομένων.
- Μπορεί να είναι ασύγχρονη: αν ο ΠΧΜΣ «κρατάει» τα δεδομένα μέχρι αυτά να παραδοθούν.
- Μια συνάρτηση περιορισμένης λήψης θα «τερματίσει» μόνο όταν τα δεδομένα ληφθούν και μπορούν να χρησιμοποιηθούν.



Περιορισμένη/Μη-περιορισμένη Αποστολή/Λήψη

Μη-περιορισμένη (Ελεύθερη) αποστολή/λήψη μηνυμάτων:

- Οι συναρτήσεις ελεύθερης αποστολής/λήψης «τερματίζουν» αμέσως χωρίς να περιμένουν καμία επικοινωνία επιβεβαίωσης.
- Οι συναρτήσεις ελεύθερης αποστολής/λήψης απλά «απαιτούν» από την βιβλιοθήκη MPI την εκτέλεση της επικοινωνίας. Ο χρήστης δεν γνωρίζει πότε αυτό θα γίνει.
- Δεν είναι ασφαλές να αλλάζουμε τον ΠΧΜΕ (δηλαδή τις μεταβλητές-δεδομένα) πριν βεβαιωθούμε ότι η εντολή αποστολής/λήψης διεκπεραιώθηκε. Υπάρχουν κατάλληλες συναρτήσεις αναμονής (wait routines) που βοηθούν σε αυτό.
- Συνήθως χρησιμοποιούνται για την πιθανή βελτίωση της απόδοσης με ταυτόχρονη επικοινωνία και εκτέλεση των διεργασιών.



Συναρτήσεις Μια-προς-μια Επικοινωνίας

- Περιορισμένη Αποστολή: **MPI_Send (buffer, count, type, dest, tag, comm)**.
- Περιορισμένη Λήψη: **MPI_Recv (buffer, count, type, source, tag, comm, status)**.
- Ελεύθερη Αποστολή: **MPI_Isend (buffer, count, type, dest, tag, comm, request)**.
- Ελεύθερη Λήψη: **MPI_Irecv (buffer, count, type, source, tag, comm, request)**.

Ορίσματα:

- ❑ **buffer:** Τα δεδομένα που θα σταλούν/ληφθούν, τα ονόματα των μεταβλητών.
- ❑ **count:** Ο αριθμός των δεδομένων.
- ❑ **type:** Ο τύπος των δεδομένων.



Τύποι Ορισμάτων της Βιβλιοθήκης MPI

MPI C Data Types	
MPI_CHAR	signed char
MPI_SHORT	signed short int
MPI_INT	signed int
MPI_LONG	signed long int
MPI_UNSIGNED_CHAR	unsigned char
MPI_UNSIGNED_SHORT	unsigned short int
MPI_UNSIGNED	unsigned int
MPI_UNSIGNED_LONG	unsigned long int
MPI_FLOAT	float
MPI_DOUBLE	double
MPI_LONG_DOUBLE	long double
MPI_BYTE	8 binary digits
MPI_PACKED	MPI_Pack(), MPI_Unpack()



Τύποι Ορισμάτων της Βιβλιοθήκης MPI

MPI Fortran Data Types	
MPI_CHARACTER	character
MPI_INTEGER	integer
MPI_REAL	real, single precision
MPI_DOUBLE_PRECISION	real, double precision
MPI_COMPLEX	complex
MPI_DOUBLE_COMPLEX	double complex
MPI_LOGICAL	logical
MPI_BYTE	8 binary digits
MPI_PACKED	MPI_Pack(), MPI_Unpack()

- Ο χρήστης μπορεί επίσης να φτιάξει τους δικούς του τύπους.
- Οι τύποι *MPI_BYTE*, *MPI_PACKED* δεν ανήκουν στους συνήθεις τύπους της C, Fortran.



Περιγραφή Ορισμάτων (συνέχεια)

- ❑ **dest:** Χρησιμοποιείται στις συναρτήσεις αποστολής και δηλώνει τον αριθμό της διεργασίας (επεξεργαστή) στην οποία θα σταλεί το μήνυμα.
- ❑ **source:** Χρησιμοποιείται στις συναρτήσεις λήψης και δηλώνει τον αριθμό της διεργασίας από την οποία θα ληφθεί το μήνυμα.
- ❑ **tag:** Αυθαίρετος θετικός ακέραιος (από 0-32767) τον οποίο θέτει ο χρήστης και είναι η ετικέτα του μηνύματος. Οι αντίστοιχες συναρτήσεις αποστολής και λήψης πρέπει να έχουν την ίδια ετικέτα.
- ❑ **comm:** Ο διαχειριστής επικοινωνίας. Προκαθορισμένος είναι ο **MPI_COMM_WORLD**.
- ❑ **status:** Για μια εντολή λήψης δηλώνει την πηγή και την ετικέτα του μηνύματος.
- ❑ **request:** Όρισμα της ελεύθερης αποστολής/λήψης. Εφόσον η ελεύθερη αποστολή/λήψη μπορεί να ολοκληρωθεί προτού δεσμευτεί ο απαιτούμενος προσωρινός χώρος μνήμης, δημιουργούμε έναν «αριθμό απαίτησης». Ο χρήστης μπορεί αργότερα να τον χρησιμοποιήσει (μέσω μιας συνάρτησης WAIT) για να ελέγξει την ολοκλήρωση της αποστολής/λήψης. Στη C το όρισμα αυτό είναι ένας δείκτης (pointer) στη δομή MPI_Request ενώ στη Fortran είναι ακέραιος.



Συναρτήσεις Περιορισμένης Αποστολής/Λήψης

- Περιορισμένη Αποστολή: **MPI_Send (buffer, count, type, dest, tag, comm)**. Πρότυπο της συνάρτησης στην C:

```
int MPI_Send(    void *      buf          /* input */,
                 int         count       /* input */,
                 MPI_Datatype type       /* input */,
                 int         dest        /* input */,
                 int         tag         /* input */,
                 MPI_Comm    comm        /* input */,
                 )
```

- Περιορισμένη Λήψη: **MPI_Recv (buffer, count, type, source, tag, comm, status)**. Πρότυπο της συνάρτησης στην C:

```
int MPI_Recv(    void *      buf          /* input */,
                 int         count       /* input */,
                 MPI_Datatype type       /* input */,
                 int         source      /* input */,
                 int         tag         /* input */,
                 MPI_Comm    comm        /* input */,
                 MPI_Status * status     /* output */
                 )
```



Παράδειγμα Χρήσης Συναρτήσεων Περιορισμένης Αποστολής/Λήψης

```
#include "mpi.h"
#include <stdio.h>

int main(int argc, char** argv)
{
    int numtasks, rank, dest, source, rc, count, tag=1;
    char inmsg[5], outmsg[]="hello";
    MPI_Status Stat;
    MPI_Init(&argc,&argv);

    MPI_Comm_size(MPI_COMM_WORLD, &numtasks);
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);

    if (rank == 0) {
        dest = 1; source = 1;
        MPI_Send(&outmsg, 5, MPI_CHAR, dest, tag, MPI_COMM_WORLD);
        MPI_Recv(&inmsg, 5, MPI_CHAR, source, tag, MPI_COMM_WORLD, &stat);
    }
    else if (rank == 1) {
        dest = 0; source = 0;
        MPI_Recv(&inmsg, 5, MPI_CHAR, source, tag, MPI_COMM_WORLD, &Stat);
        MPI_Send(&outmsg, 5, MPI_CHAR, dest, tag, MPI_COMM_WORLD);
    }
    MPI_Get_count(&Stat, MPI_CHAR, &count);
    printf("Task %d: Received %d char(s) from task %d with tag %d \n", rank, count, Stat.MPI_SOURCE,
        Stat.MPI_TAG);
    MPI_Finalize();
}
```



Συναρτήσεις Ελεύθερης Αποστολής/Λήψης

- Ελεύθερη Αποστολή: **MPI_Isend (buffer, count, type, dest, tag, comm, request)**.
Πρότυπο της συνάρτησης στην C:

```
int MPI_Isend(    void *      buf          /* input */,
                 int         count       /* input */,
                 MPI_Datatype type       /* input */,
                 int         dest        /* input */,
                 int         tag         /* input */,
                 MPI_Comm    comm        /* input */,
                 MPI_Request* request    /* output */,
                 )
```

- Ελεύθερη Λήψη: **MPI_Irecv (buffer, count, type, source, tag, comm, request)**.
Πρότυπο της συνάρτησης στην C:

```
int MPI_Irecv(   void *      buf          /* input */,
                 int         count       /* input */,
                 MPI_Datatype type       /* input */,
                 int         source      /* input */,
                 int         tag         /* input */,
                 MPI_Comm    comm        /* input */,
                 MPI_Request* request    /* output */,
                 )
```



Παράδειγμα Χρήσης Συναρτήσεων Ελεύθερης Αποστολής/Λήψης

```
program ringtopo

include 'mpif.h'
integer numtasks, rank, next, prev, buf(2), tag1, tag2, ierr

integer stats(MPI_STATUS_SIZE,4), reqs(4)
tag1 = 1; tag2 = 2

call MPI_INIT(ierr)
call MPI_COMM_RANK(MPI_COMM_WORLD, rank, ierr)
call MPI_COMM_SIZE(MPI_COMM_WORLD, numtasks, ierr)

prev = rank - 1; next = rank + 1
if (rank == 0) prev = numtasks - 1
if (rank == numtasks - 1) next = 0

call MPI_Irecv(buf(1), 1, MPI_INTEGER, prev, tag1, MPI_COMM_WORLD, reqs(1), ierr)
call MPI_Irecv(buf(2), 1, MPI_INTEGER, next, tag2, MPI_COMM_WORLD, reqs(2), ierr)
call MPI_Isend(rank, 1, MPI_INTEGER, prev, tag2, MPI_COMM_WORLD, reqs(3), ierr)
call MPI_Isend(rank, 1, MPI_INTEGER, next, tag1, MPI_COMM_WORLD, reqs(4), ierr)

call MPI_WAITALL(4, reqs, stats, ierr)
call MPI_FINALIZE(ierr)

stop
end
```




Συλλογική Επικοινωνία Collective Communication

- Η συλλογική επικοινωνία περιλαμβάνει αποστολή / λήψη μηνυμάτων μεταξύ όλων των διεργασιών σε ένα διαχειριστή επικοινωνίας.
- Διάφοροι τύποι συναρτήσεων-υποπρογραμμάτων συλλογικής επικοινωνίας:
 - Συγχρονισμένη επικοινωνία (**Synchronization**): οι διεργασίες περιμένουν έως όλη η επικοινωνία βρεθεί στο σημείο συγχρονισμού.
 - Μεταφορά Δεδομένων (**Data Movement**): εκπομπή (broadcast), διασπορά/συγκέντρωση (scatter/gather), όλοι σε όλους (all to all).
 - Συλλογική υπολογισμοί – περιστολή (**reduction**): μια διεργασία συλλέγει δεδομένα από όλες τις άλλες και εκτελεί μια πράξη (π.χ. πρόσθεση, πολλαπλασιασμός, κλπ.) σε αυτά τα δεδομένα.
- Οι συλλογική επικοινωνία είναι περιορισμένου (blocking) τύπου.
- Υπάρχουν επίσης συναρτήσεις παρακολούθησης και ελέγχου αποστολής/λήψης μηνυμάτων.



Υποπρογράμματα Συλλογικής Επικοινωνίας MPI

- **MPI_Barrier:** Δημιουργεί ένα σημείο συγχρονισμού. Κάθε διεργασία που φτάνει στο σημείο που το MPI_Barrier καλείται, περιμένει ως ότου όλες οι διεργασίες φτάσουν σε αυτό το σημείο

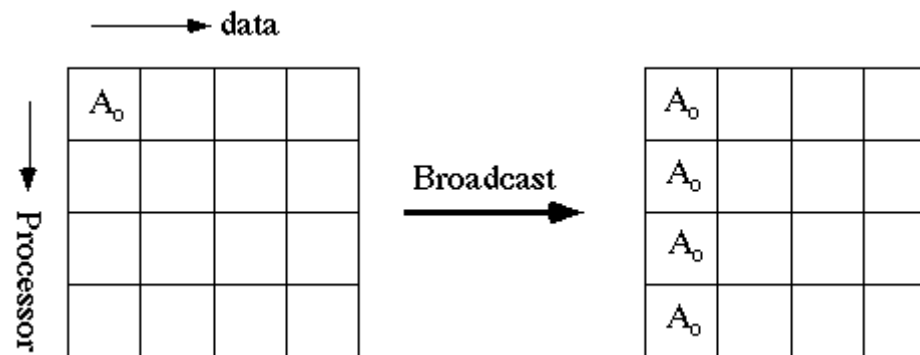
C: *MPI_Barrier(comm);*

Fortran: *MPI_BARRIER(comm, ierr)*

- **MPI_Bcast:** Εκπέμπει ένα μήνυμα από την διεργασία με αριθμό “root” σε όλες τις άλλες διεργασίες μέσα στον Δ.Ε.

C: *MPI_Bcast(&buffer, count, type, root, comm);*

Fortran: *MPI_BCAST(buffer, count, type, root, comm, ierr)*





Υποπρογράμματα Συλλογικής Επικοινωνίας MPI

- **MPI_Scatter:** Διασκορπίζει ένα μήνυμα από την διεργασία με αριθμό “root” σε όλες τις άλλες διεργασίες μέσα στον Δ.Ε.

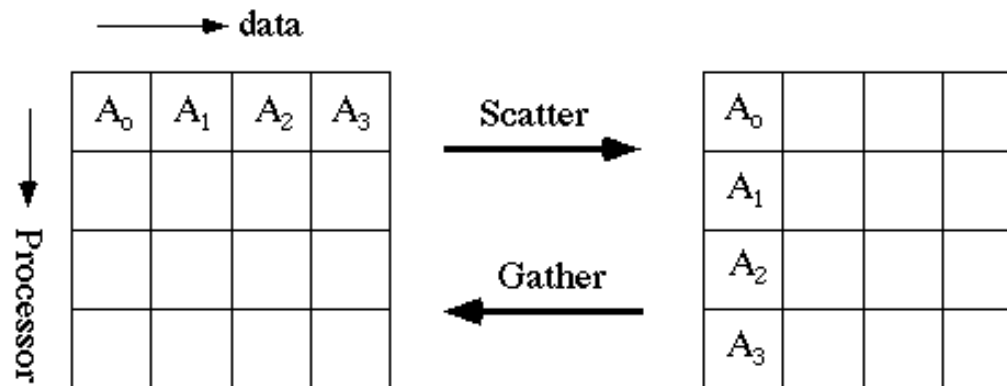
C: *MPI_Scatter (&sendbuf, sendcnt, sendtype, &recvbuf, recv, recvtype, root, comm);*

Fortran: *MPI_SCATTER(sendbuf, sendcnt, sendtype, recvbuf, recv, recvtype, root, comm, ierr)*

- **MPI_Gather:** Μαζεύει ένα μήνυμα από κάθε διεργασία στη διεργασία “root”.

C: *MPI_Gather (&sendbuf, sendcnt, sendtype, &recvbuf, recv, recvtype, root, comm);*

Fortran: *MPI_SCATTER(sendbuf, sendcnt, sendtype, recvbuf, recv, recvtype, root, comm, ierr)*





Υποπρογράμματα Συλλογικής Επικοινωνίας MPI

- **MPI_Reduce:** Εφαρμόζει μια πράξη περιστολής σε όλα τα μέλη του Δ.Ε. και στέλνει το αποτέλεσμα στη διεργασία με αριθμό “root”.

C: *MPI_Reduce (&buffer, count, type, op, root, comm);*

Fortran: *call MPI_REDUCE(buffer, count, type, op, root, comm, ierr)*

Οι σημαντικότερες προκαθορισμένες πράξεις τύπου MPI_Reduce είναι:

MPI Reduction Operation		C Data Types	Fortran Data Types
MPI_MAX	maximum	integer, float	integer, real, complex
MPI_MIN	minimum	integer, float	integer, real, complex
MPI_SUM	sum	integer, float	integer, real, complex
MPI_PROD	product	integer, float	integer, real, complex
MPI LAND	logical AND	integer	logical
MPI_MAXLOC	max value and location	float, double, long double	real, complex, double precision
MPI_MINLOC	min value and location	float, double, long double	real, complex, double precision



Παράδειγμα Χρήσης Υποπρογραμμάτων Συλλογικής Επικοινωνίας

```
#include "mpi.h"
#include <stdio.h>
#define SIZE 4

int main(int argc, char** argv)
{
    int numtasks, rank, sendcount, recvcount, source;
    float sendbuf[SIZE][SIZE] = { {1.0, 2.0, 3.0, 4.0}, {5.0, 6.0, 7.0, 8.0}, {9.0, 10.0, 11.0, 12.0}, {13.0, 14.0, 15.0, 16.0} };
    float recvbuf[SIZE];

    MPI_Init(&argc,&argv);
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
    MPI_Comm_size(MPI_COMM_WORLD, &numtasks);

    if (numtasks == SIZE) {
        source = 1;
        sendcount = SIZE;
        recvcount = SIZE;
        MPI_Scatter(sendbuf,sendcount,MPI_FLOAT,recvbuf,recvcount,MPI_FLOAT,source,MPI_COMM_WORLD);
        printf("rank= %d Results: %f %f %f %f\n",rank,recvbuf[0],recvbuf[1],recvbuf[2],recvbuf[3]);
    }
    else
        printf("Must specify %d processors. Terminating.\n",SIZE);

    MPI_Finalize();
}
```



Derived Datatypes

- Τα βασικά ορίσματα της βιβλιοθήκης MPI που είδαμε παραπάνω (**primitive MPI Datatypes**) επιτρέπουν μεταφορά απλών δεδομένων ή σύνολο μεταβλητών του ίδιου τύπου οι οποίες είναι συνεχείς (**contiguous**) στη μνήμη.
- **Derived Datatypes** είναι πιο πολύπλοκα ορίσματα που μπορούν να εμπεριέχουν μεταβλητές διαφορετικού είδους ή μεταβλητές που δεν είναι συνεχείς στη μνήμη.
- Τα ορίσματα MPI Derived Datatypes είναι ένα σύνολο μεταβλητών η καθεμία από τις οποίες έχει ως τύπο ένα οποιοδήποτε primitive MPI Datatype.
- **Λόγοι χρήσης:** Είναι ο μόνος τρόπος να στείλουμε με μια πράξη επικοινωνίας (μια αποστολή/λήψη) μια δομή (structure), μέρος ενός πίνακα (π.χ. Sub-block of a matrix), ή ένα σύνολο δεδομένων διαφορετικού τύπου (π.χ. int. and float numbers).
- Υπάρχουν διαφορετικοί τρόποι ορισμού ενός MPI derived datatype:
 - Contiguous
 - Vector
 - Indexed
 - Struct



Υποπρογράμματα-Συναρτήσεις MPI Derived Datatypes

-- Βασικά υποπρογράμματα δημιουργίας ορισμάτων MPI derived datatype:

- **MPI_Type_contiguous:** ο απλούστερος τρόπος. Δημιουργεί ένα καινούριο όρισμα αντιγράφοντας τα ήδη υπάρχοντα ορίσματα.

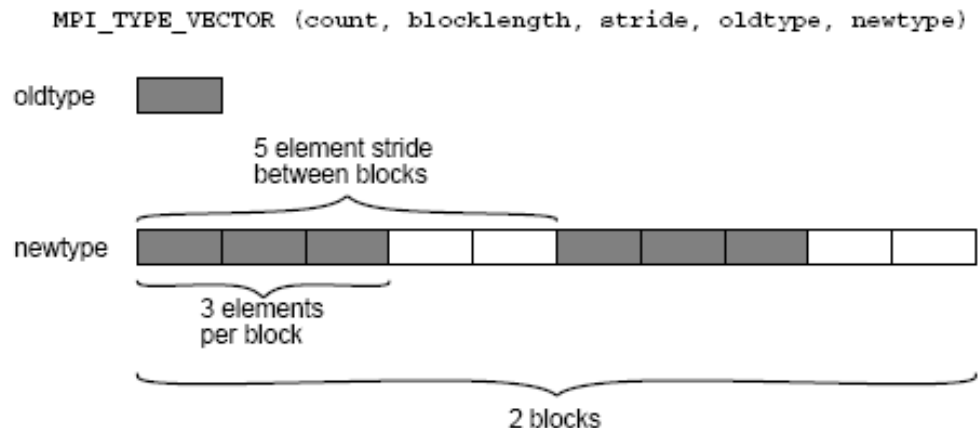
C: *MPI_Type_contiguous (count, oldtype, &newtype)*

Fortran: *call MPI_TYPE_CONTIGUOUS (count, oldtype, newtype, ierr)*

- **MPI_Type_vector:** παρόμοιο με το contiguous αλλά επιτρέπει κενά (stride) στις μετακινήσεις (displacements) των ορισμάτων.

C: *MPI_Type_vector (count, blocklength, stride, oldtype, &newtype)*

Fortran: *call MPI_TYPE_VECTOR (count, blocklength, stride, oldtype, newtype, ierr)*



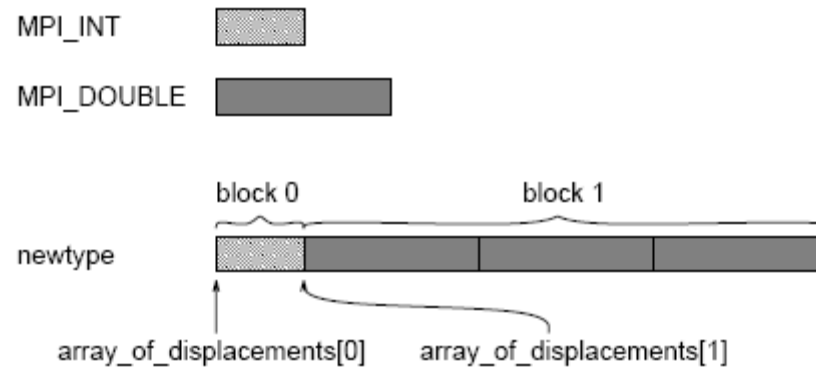


Υποπρογράμματα-Συναρτήσεις MPI Derived Datatypes

- **MPI_Type_struct:** ο πιο ευέλικτος τρόπος. Ο χρήστης ελέγχει πλήρως τη δομή του νέου ορίσματος..

C: *MPI_Type_struct* (*count*, *blocklens*[], *offsets*[], *old_types*, &*newtype*)

Fortran: *call MPI_TYPE_STRUCT* (*count*,*blocklens*[],*offsets*[],*old_types*, *newtype*, *ierr*)



- **MPI_Type_Extend:** Επιστρέφει τον μέγεθος (σε bytes) του derived datatype ορίσματος.

C: *MPI_Comm_rank* (*datatype*, &*extent*);

Fortran: *call MPI_COMM_RANK*(*datatype*, *extent*, *ierr*)



Derived Datatypes

- Κατασκευή ενός ορίσματος derived datatype:

A) Ορισμός του ορίσματος χρησιμοποιώντας μια από τις παραπάνω υπορουτίνες (MPI_Type_contiguous, MPI_Type_vector, MPI_Type_struct, ...).

B) Παράδοση του νέου ορίσματος. Το καινούριο όρισμα «παραδίδεται» καλώντας την:
MPI_Type_commit(&newtype)

- Επίσης πολύ συχνά είναι χρήσιμο να ελευθερώνετε η μνήμη από τα καινούρια ορίσματα όταν αυτά δεν χρειάζονται. Αυτό γίνεται καλώντας την ρουτίνα:

MPI_Type_free(&newtype)



Τοπολογία Διεργασιών Τύπου MPI

- Ο παράλληλος προγραμματισμός τύπου MPI επιτρέπει την αποτύπωση/διευθέτηση (**mapping/ordering**) των διεργασιών σε ένα γεωμετρικό σχήμα.
- Οι δύο κυριότερες τοπολογίες είναι: Η Καρτεσιανή (Grid) και η Γραφική (Graph).
- Οι τοπολογίες διεργασιών MPI είναι εικονικές: δεν υπάρχει σχέση με την τοπολογία των επεξεργαστών.
- Λόγοι χρήσης:
 - Ευκολία: Μπορεί να είναι χρήσιμες για προβλήματα με συγκεκριμένα προφίλ επικοινωνίας, π.χ. η Καρτεσιανή τοπολογία είναι κατάλληλη για προβλήματα όπου απαιτείται επικοινωνία μιας διεργασίας με της 4 γειτονικές της σε μορφή grid
 - Αποδοτικότητα: μια συγκεκριμένη τοπολογία μπορεί να βελτιστοποιήσει τις διεργασίες με βάση την αρχιτεκτονική του παράλληλου συστήματος.

- Παράδειγμα (4x4 Καρτεσιανή τοπολογία):

0 (0,0)	1 (0,1)	2 (0,2)	3 (0,3)
4 (1,0)	5 (1,1)	6 (1,2)	7 (1,3)
8 (2,0)	9 (2,1)	10 (2,2)	11 (2,3)
12 (3,0)	13 (3,1)	14 (3,2)	15 (3,3)



Παράδειγμα Δημιουργίας Καρτεσιανής Τοπολογίας

```
program cartesian
include 'mpif.h'

integer SIZE, UP, DOWN, LEFT, RIGHT
parameter(SIZE=16, UP=1, DOWN=2, LEFT=3, RIGHT=4)
integer numtasks, rank, source, dest, outbuf, i, tag, ierr, inbuf(4), nbrs(4), dims(2), coords(2),
integer stats(MPI_STATUS_SIZE, 8), reqs(8), cartcomm, periods(2), reorder
data inbuf /MPI_PROC_NULL,MPI_PROC_NULL,MPI_PROC_NULL,MPI_PROC_NULL/
data dims /4,4/, tag /1/, periods /0,0/, reorder /0/

call MPI_INIT(ierr)
call MPI_COMM_SIZE(MPI_COMM_WORLD, numtasks, ierr)

if (numtasks== SIZE) then
call MPI_CART_CREATE(MPI_COMM_WORLD, 2, dims, periods, reorder, cartcomm, ierr)
call MPI_COMM_RANK(cartcomm, rank, ierr)
call MPI_CART_COORDS(cartcomm, rank, 2, coords, ierr)
print *, 'rank= ',rank,'coords= ',coords
call MPI_CART_SHIFT(cartcomm, 0, 1, nbrs(UP), nbrs(DOWN), ierr)
call MPI_CART_SHIFT(cartcomm, 1, 1, nbrs(LEFT), nbrs(RIGHT), ierr)
outbuf = rank
do i=1,4
dest = nbrs(i)
source = nbrs(i)
call MPI_ISEND(outbuf, 1, MPI_INTEGER, dest, tag, MPI_COMM_WORLD, reqs(i), ierr)
call MPI_IRECV(inbuf(i), 1, MPI_INTEGER, source, tag, MPI_COMM_WORLD, reqs(i+4), ierr)
end do
call MPI_WAITALL(8, reqs, stats, ierr)
print *, 'rank= ',rank,' coords= ',coords, ' neighbors(u,d,l,r)= ',nbrs print *, 'rank= ',rank,' ', ' inbuf(u,d,l,r)= ',inbuf
else
print *, 'Must specify',SIZE,' processors. Terminating.'
end if
call MPI_FINALIZE(ierr)

end
```



Βιβλιογραφία

- *Parallel Programming*, B. Wilkinson, M. Allen, Prentice Hall, 2nd Ed. 2005.
- *Introduction to Parallel Computing*, A. Grama, G. Karypis, V. Kumar, A. Gupta, Addison-Wesley, 2003.
- *Designing and Building Parallel Programs*, Ian Foster, Addison-Wesley 1994.
- *Parallel Computing: Theory and Practice*, M. J. Quinn, McGraw-Hill, 1994.
- MPI: <http://www.mpi-forum.org/>
- <http://www.epcc.ed.ac.uk/library/documentation/training/>: On-line courses include MPI, HPF, Mesh Generation, Introduction to Computational Science, HPC in Business.