



# Συστήματα Παράλληλης & Κατανεμημένης Επεξεργασίας

**Ενότητα 4:** MPI\_ANY\_TAG, MPI\_ANY\_SOURCE, MPI\_Bcast,  
MPI\_Wtime, MPI\_Wait, MPI\_Test, MPI\_Scatter

Δρ. Μηνάς Δασυγένης

[mdasyg@ieee.org](mailto:mdasyg@ieee.org)

Εργαστήριο Ψηφιακών Συστημάτων και Αρχιτεκτονικής Υπολογιστών

<http://arch.ict.e.uowm.gr/mdasyg>

Τμήμα Μηχανικών Πληροφορικής και Τηλεπικοινωνιών

---



# Άδειες Χρήσης

---

- Το παρόν εκπαιδευτικό υλικό υπόκειται σε άδειες χρήσης Creative Commons.
- Για εκπαιδευτικό υλικό, όπως εικόνες, που υπόκειται σε άλλου τύπου άδειας χρήσης, η άδεια χρήσης αναφέρεται ρητώς.



# Χρηματοδότηση

- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στα πλαίσια του εκπαιδευτικού έργου του διδάσκοντα.
- Το έργο «**Ανοικτά Ψηφιακά Μαθήματα στο Πανεπιστήμιο Δυτικής Μακεδονίας**» έχει χρηματοδοτήσει μόνο τη αναδιαμόρφωση του εκπαιδευτικού υλικού.
- Το έργο υλοποιείται στο πλαίσιο του Επιχειρησιακού Προγράμματος «Εκπαίδευση και Δια Βίου Μάθηση» και συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικούς πόρους.



Ευρωπαϊκή Ένωση  
Ευρωπαϊκό Κοινωνικό Ταμείο



ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ  
ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ  
*επένδυση στην κοινωνία της γνώσης*  
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ ΚΑΙ ΘΡΗΣΚΕΥΜΑΤΩΝ  
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ

Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης



ΕΣΠΑ  
2007-2013  
Πρόγραμμα για την ανάπτυξη  
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ



# Σκοπός της Ενότητας

---

- Η παρουσίαση βασικών δομών επικοινωνίας με το MPI.



# Το tag στις συναρτήσεις send, recv

---

- Σε αυτές τις συναρτήσεις η πέμπτη παράμετρος είναι το tag της επικοινωνίας.
- Το TAG είναι ένας αριθμός (π.χ. το 0).
- Γενικά το tag στην αποστολή και στη λήψη θα πρέπει να είναι το ίδιο.
- Μπορούμε όμως αντί για να δώσουμε έναν αριθμό να χρησιμοποιούμε την παράμετρο (**μόνο recv**) **MPI\_ANY\_TAG** η οποία δεν ορίζει κάποιο συγκεκριμένο tag.
- Μπορεί να χρησιμοποιηθεί στην λήψη (σε κάποιες υλοποιήσεις openMPI χρησιμοποιείται και στο send).



# Λήψη μηνυμάτων από οποιαδήποτε διεργασία

---

- Το τέταρτο όρισμα της `recv` (λήψης μηνυμάτων) ορίζει από ποια διεργασία περιμένει να λάβει μήνυμα.
- Είναι ένας αριθμός που υποδηλώνει το `rank` (π.χ. 5).
- Αν δε μας ενδιαφέρει να ορίσουμε ακριβώς τη διεργασία που θα στείλει το μήνυμα, μπορούμε αντί για τον αριθμό να χρησιμοποιήσουμε τη μεταβλητή: **`MPI_ANY_SOURCE`**
- Χρησιμοποιείται μόνο στη `recv`.



# Το όρισμα count

---

- Είναι η 2η παράμετρος της recv.
- Δεν αναφέρεται στο ακριβές πλήθος των στοιχείων, αλλά στο μέγιστο πλήθος των στοιχείων που αναμένουμε.
- Είναι δυνατό να είναι 0:
  - Ο αποστολέας στέλνει μόνο μια επικεφαλίδα χωρίς δεδομένα.



# Η συνάρτηση MPI\_Bcast (1/3)

---

- Αποστολή μηνύματος σε όλες τις διεργασίες (και στον εαυτό της):
  - **MPI\_Bcast (void \* buffer, int count, MPI\_Datatype dataType, int srcRank, MPI\_Comm comm)**
    - void \* buffer, pointer στο μήνυμα που θα σταλεί.
    - int count, πλήθος στοιχείων που θα σταλούν.
    - MPI\_Datatype dataType, τύπος στοιχείων.
    - int srcRank, ποια διεργασία είναι αυτή που το στέλνει.
    - MPI\_Comm comm σε ποιο COMMUNICATOR θα σταλεί.





# Η συνάρτηση MPI\_Bcast (2/3)

---

- **ΠΡΟΣΟΧΗ (1):** Δεν απαιτείται η χρήση της `recv` για λήψη του μηνύματος.
- **ΠΡΟΣΟΧΗ (2):** Για να λάβει σωστά το μήνυμα μια άλλη διεργασία θα πρέπει να χρησιμοποιήσει τη συνάρτηση **MPI\_Bcast** με τα ίδια ακριβώς ορίσματα που έχει χρησιμοποιήσει η αρχική συνάρτηση.
- Μετά τον τερματισμό της συνάρτησης όλες οι ενδιάμεσες μνήμες των διεργασιών του **COMMUNICATOR** έχουν τα ίδια περιεχόμενα.
- Αυτό γίνεται άμεσα και δε χρειάζεται παρεμβολή από τον χρήστη.



# Η συνάρτηση MPI\_Bcast (3/3)

---

- Η MPI\_Bcast μπορεί να αντικατασταθεί με ένα βρόχο για όλα τα rank ως εξής:

```
for counter=0;counter<size;counter++)  
MPI_Send(&a,1,MPI_DOUBLE,counter,0,  
MPI_COMM_WORLD); (προσοχή next slide)
```

- Κάθε rank θα έπρεπε να χρησιμοποιήσει την recv για να λάβει το μήνυμα.

```
MPI_Recv(&a,1,MPI_DOUBLE,0,MPI_ANY_TAG,  
MPI_COMM_WORLD);
```

- Η MPI\_Bcast είναι πιο γρήγορη και καλύτερη.



# Αποστολή μηνύματος στον εαυτό (send-to-itself)

---

- Στην προηγούμενη διαφάνεια η MPI\_Send επειδή εκτελούνται για όλο το communicator size έστειλε ΚΑΙ στον εαυτό της.
- Αν και τις περισσότερες φορές θα λειτουργήσει χωρίς να κάνει block, θα πρέπει να αποφεύγεται:

**"Therefore, it is unsafe and non-portable to send self-messages with the standard mode, blocking send and receive operations described so far, since this may lead to deadlock.«**

- The book "MPI: The complete reference, Vol. 1, 2nd Ed.", by Snir et al., page 42, section 2.9.9.



# Η συνάρτηση MPI\_Wtime()

---

- Χρησιμοποιείται για να μετρήσει το χρόνο εκτέλεσης ενός τμήματος κώδικα.
- Πρότυπο:

**double MPI\_Wtime(void)**

- Μπορεί να χρησιμοποιηθεί ώστε να μας δώσει πληροφορίες για την χρονική στιγμή έναρξης ή λήξης ενός προγράμματος. Αν καλέσουμε τη συνάρτηση αυτή στην αρχή μιας διαδικασίας (π.χ. μετά την MPI\_Comm\_size) και μια στο τέλος (πριν την MPI\_Finalize), τότε η διαφορά τους θα μας δώσει τον χρόνο εκτέλεσης του προγράμματος σε δευτερόλεπτα.



# Παράδειγμα MPI\_Wtime()

---

```
double startwtime, endwtime;  
startwtime = MPI_Wtime();  
(.....κομμάτι κώδικα.....)  
endwtime = MPI_Wtime();  
printf ("wall clock time = %f \n",  
endwtime-startwtime);
```



# Άλλες ενδιαφέρουσες ρουτίνες

---

- **MPI\_Wait**(MPI\_Request \*req, MPI\_Status \*status).
  - Περιμένει έως ότου η επικοινωνία αναφοράς req λήξει.
- **MPI\_Test**(MPI\_Request \*req, int \*flag, MPI\_Status \*status).
  - Επιστρέφει flag=TRUE εφόσον η επικοινωνία αναφοράς req έχει λήξει επιτυχώς και flag=FALSE σε άλλη περίπτωση.



# Συλλογικές κλήσεις

---

- Κάθε διεργασία εκτελεί τις ίδιες ρουτίνες επικοινωνίας με τις υπόλοιπες.
- Δεν χρησιμοποιούνται tags αλλά communicators.
- Δεν υπάρχουν non-blocking συλλογικές κλήσεις.
- Τρεις τύποι:
  - Συγχρονισμός.
  - Μετακίνηση δεδομένων.
  - Συλλογικοί υπολογισμοί.



# Παραδείγματα συλλογικών κλήσεων

---

- **MPI\_Barrier**(MPI\_Comm communicator).
  - παγώνει όλες τις διεργασίες έως ότου αυτές φτάσουν στο συγκεκριμένο σημείο ελέγχου. Όταν όλες οι διεργασίες εκτελέσουν τη MPI\_Barrier τότε συνεχίζεται η εκτέλεση όλων κανονικά.
- **MPI\_Bcast**(void \*buf, int count, MPI\_Datatype data\_type, int root, MPI\_Comm comm).
  - Σήμα one-to-all (αναπτύχθηκε σε προηγούμενη παρουσίαση).





# Η συνάρτηση MPI\_Scatter() (1/5)

---

- Συνάρτηση διαμοιρασμού δεδομένων σε όλες τις διεργασίες της ομάδας.
- Η συνάρτηση MPI\_Scatter ανήκει στην κατηγορία των συναρτήσεων που πραγματοποιούν συλλογικές επικοινωνίες (collective communications). Το βασικό χαρακτηριστικό των συναρτήσεων συλλογικής επικοινωνίας είναι ότι κάποια διεργασία ανταλλάσσει δεδομένα προς όλες τις υπόλοιπες διεργασίες.

```
int MPI_Scatter (void * sendBuf, int sendCount, MPI_Datatype  
sendType, void * recvBuf, int recvCount, MPI_Datatype recvType,  
int root, MPI_Comm comm);
```



# Η συνάρτηση MPI\_Scatter() (2/5)

---

- **sendBuf:** Περιοχή μνήμης που είναι τα δεδομένα προς διασπορά.
- **sendCount:** Πλήθος δεδομένων προς διασπορά **ανά διεργασία.**
- **sendType:** Τύπος δεδομένων προς διασπορά.
- **recvBuf:** Περιοχή μνήμης που θα αποθηκευθούν τα δεδομένα από την διεργασία παραλήπτη.
- **recvCount:** Πλήθος δεδομένων που παραλαμβάνονται ανά διεργασία.
- **recvType:** Τύπος δεδομένων που παραλαμβάνονται.
- **root:** Η διεργασία ρίζα που πρόκειται να διασπείρει τα δεδομένα.
- **comm:** Ο χειριστής στον οποίο ανήκουν οι διεργασίες αποστολής (MPI\_COMM\_WORLD).



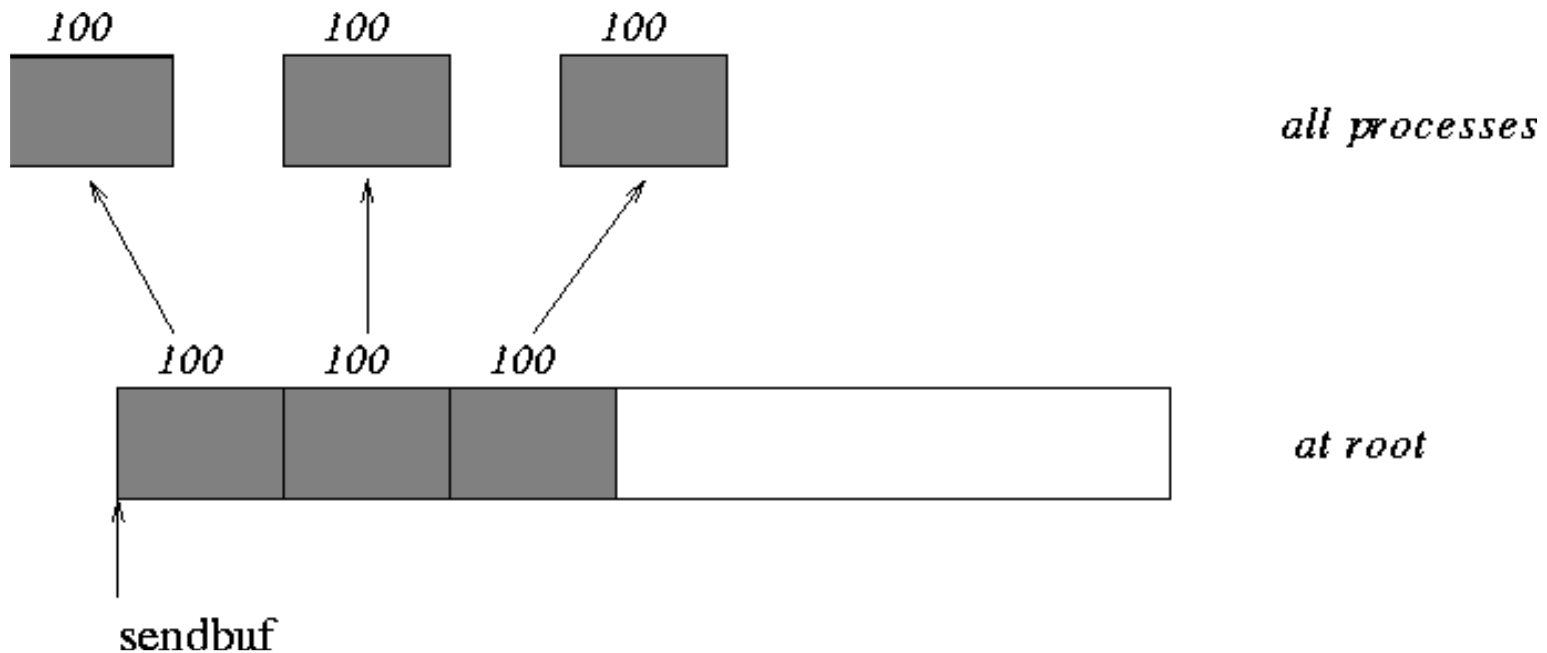
# Η συνάρτηση MPI\_Scatter() (3/5)

- Συγκεκριμένα, η MPI\_Scatter κάνει διασπορά δεδομένων (Scattering), όπου τα δεδομένα μίας διεργασίας μοιράζονται με κατάλληλο τρόπο σε όλες τις υπόλοιπες καθώς και στον εαυτό της. Δεν απαιτείται να ξεκινήσει η κάθε διεργασία-παραλήπτης μια συνάρτηση MPI\_Recv για την παραλαβή των δεδομένων.
- Παράδειγμα: Έστω ένας πίνακας: A με 300 ακεραίους και 3 διεργασίες, η κλήση της συνάρτησης:  
**MPI\_Scatter (&A, 100, MPI\_INT, &B, 100, MPI\_INT, 0, MPI\_COMM\_WORLD);**
- Θα έχει σαν αποτέλεσμα η διεργασία 0 να διασπείρει τα δεδομένα του πίνακα A, οι 3 διεργασίες θα παραλάβουν στον πίνακα B: τα 100 πρώτα στοιχεία η διεργασία 0, τα 100 επόμενα η διεργασία 1 και τα 100 τελευταία η διεργασία 2.



# Η συνάρτηση MPI\_Scatter() (4/5)

- Η συνάρτηση διαμοιράζει τα δεδομένα σε όλες τις διεργασίες της ομάδας.



# Η συνάρτηση MPI\_Scatter() (5/5)

---

- Προσοχή:
  - Μπορούν να διαμοιραστούν ΜΟΝΟ ομάδες με διακριτά στοιχεία Για παράδειγμα δε μπορούμε να πούμε να διαμοιράσουμε τους πραγματικούς αριθμούς από 0 έως 1 σε  $N$  διεργασίες.
  - Αν όμως είχαμε έναν πίνακα με πραγματικούς αριθμούς θα μπορούσε να διαμοιραστεί.



---

# Τέλος Ενότητας



Ευρωπαϊκή Ένωση  
Ευρωπαϊκό Κοινωνικό Ταμείο



Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

