

# Μονόπλευρες επικοινωνίες στο MPI-2

Γιώργος Καππές

A.M.: 235

## Περίληψη

Οι μονόπλευρες επικοινωνίες που παρέχει το MPI έχουν ως στόχο να παρέχουν τη ευκολία της άμεσης προσπέλασης απομακρυσμένων μνημών, καθώς και τη δυνατότητα για υψηλότερες επιδόσεις από αυτές που παρέχουν οι επικοινωνίες σημείου προς σημείο, ειδικότερα σε δίκτυα τα οποία υποστηρίζουν εγγενώς τις μονόπλευρες επικοινωνίες, όπως τα Myrinet και InfiniBand. Οι μονόπλευρες επικοινωνίες αποτελούν ένα εναλλακτικό μοντέλο επικοινωνίας στο οποίο όλες οι παράμετροι που χαρακτηρίζουν τη λειτουργία του συστήματος μπορούν να καθοριστούν μόνο από το ένα άκρο της επικοινωνίας. Σε αυτό το μοντέλο, η διεργασία που περιέχει τα δεδομένα δε γνωρίζει τις ταυτότητες των υπόλοιπων διεργασιών που τα προσπελούν και έτσι, δε συμμετέχει στην ανταλλαγή δεδομένων. Αντίθετα, οι διεργασίες που χρειάζονται τα δεδομένα τα προσπελούν από τη διεργασία που τα κατέχει μέσω ειδικών συναρτήσεων που επιτελούν λειτουργίες εγγραφής και ανάγνωσης δεδομένων που βρίσκονται σε απομακρυσμένες μνήμες (Remote Memory Access).

Σε αυτόν τον οδηγό αρχικά θα περιγράψουμε σύντομα τα βασικά χαρακτηριστικά των μονόπλευρων επικοινωνιών. Στη συνέχεια θα αναφέρουμε τις συναρτήσεις που παρέχει το MPI για μονόπλευρες επικοινωνίες και θα περιγράψουμε τον τρόπο με τον οποίο χρησιμοποιούνται.

## 1 Εισαγωγή

Σε πολλές παράλληλες επιστημονικές εφαρμογές η κατανομή των δεδομένων μπορεί να αλλάζει δυναμικά και τα πρότυπα πρόσβασης στα δεδομένα μπορεί να είναι γενικά ακανόνιστα. Γι' αυτά τα είδη των εφαρμογών, η διεργασία που περιέχει τα δεδομένα δε γνωρίζει ούτε τις ταυτότητες των διεργασιών που τα προσπελούν, αλλά ούτε και τον τρόπο με τον οποίο γίνεται αυτή η προσπέλαση. Έτσι, δεν είναι δυνατή η ανταλλαγή δεδομένων ανάμεσά τους με χρήση επικοινωνιών που βασίζονται στο μοντέλο σημείο προς σημείο. Για να καταστεί δυνατή η επικοινωνία σε τέτοιες περιπτώσεις οι παράμετροι που χαρακτηρίζουν την επικοινωνία καθορίζονται μόνο από το ένα άκρο της επικοινωνίας, δηλαδή μόνο από τις διεργασίες που προσπελούν τα δεδομένα.

Το πρότυπο MPI-2 παρέχει μια διεπαφή και ένα προγραμματιστικό μοντέλο που επιτρέπει την ανάπτυξη εφαρμογών οι οποίες στηρίζονται στις μονόπλευρες επικοινωνίες. Σε αυτό το μοντέλο, οι διεργασίες μπορούν να προσπελάσουν απευθείας τα δεδομένα μιας άλλης διεργασίας. Το μοντέλο αυτό αναφέρεται και ως απομακρυσμένη προσπέλαση μνήμης (Remote Memory Access).

## **1.1 Παράθυρο μνήμης και περίοδοι επικοινωνίας**

Στο μοντέλο των μονόπλευρων επικοινωνιών η διεργασία που πραγματοποιεί τη διαδικασία της απομακρυσμένης προσπέλασης των δεδομένων ορίζεται ως προέλευση (origin) και η διεργασία της οποίας η μνήμη προσπελάνεται ορίζεται ως στόχος (target). Η περιοχή μνήμης της διεργασίας στόχου, την οποία η διεργασία προέλευσης μπορεί να προσπελάσει, ονομάζεται παράθυρο. Η δημιουργία ενός παραθύρου είναι μια συλλογική λειτουργία και όλες οι λειτουργίες επικοινωνίας και συγχρονισμού σχετίζονται με ένα συγκεκριμένο παράθυρο. Οι απομακρυσμένες διευθύνσεις όπου βρίσκονται αποθηκευμένα τα δεδομένα καθορίζονται ως offsets από τη βάση του παραθύρου και έτσι, οι διεργασίες που τα προσπελάζουν δε χρειάζεται να γνωρίζουν την ακριβή απομακρυσμένη διεύθυνση μνήμης όπου είναι τοποθετημένα.

Οι διεργασίες προέλευσης δε μπορούν να προσπελάσουν οποιαδήποτε στιγμή το παράθυρο μνήμης μιας διεργασίας στόχου. Το MPI χρησιμοποιεί την έννοια της περιόδου (epoch) για να καθορίσει την περίοδο κατά την οποία αυτή η προσπέλαση είναι εφικτή. Στο πλαίσιο αυτό, το χρονικό διάστημα κατά το οποίο το απομακρυσμένο παράθυρο μνήμης της διεργασίας στόχου παραμένει ανοιχτό ονομάζεται ως περίοδος έκθεσης (exposure epoch). Κατά τη διάρκεια αυτού του χρονικού διαστήματος, η διεργασία προέλευσης μπορεί να προσπελάσει τα δεδομένα που βρίσκονται σε αυτό το παράθυρο. Επιπρόσθετα, θα πρέπει και η διεργασία προέλευσης να ορίσει το δικό της χρονικό διάστημα κατά το οποίο θα προσπελάσει το παράθυρο μνήμης της διεργασίας στόχου. Αυτό το χρονικό διάστημα ονομάζεται περίοδος προσπέλασης (access epoch). Μόλις η διεργασία προέλευσης ολοκληρώσει την προσπέλαση του απομακρυσμένου παραθύρου μνήμης τερματίζει την περίοδο προσπέλασης, ενώ η διεργασία στόχος τερματίζει την περίοδο έκθεσης. Από τη στιγμή αυτή και έπειτα καμία διεργασία δε μπορεί να προσπελάσει το παράθυρο μνήμης της διεργασίας στόχου.

## **1.2 Λειτουργίες επικοινωνίας**

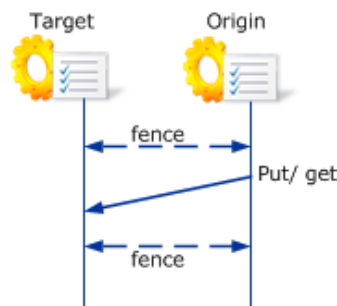
Το MPI-2 καθορίζει διάφορες λειτουργίες επικοινωνίας οι οποίες μπορούν να χρησιμοποιηθούν στις μονόπλευρες επικοινωνίες. Σε αυτές τις λειτουργίες ανήκουν η απομακρυσμένη εγγραφή (MPI\_Put), η απομακρυσμένη ανάγνωση (MPI\_Get) και η απομακρυσμένη

ενημέρωση δεδομένων (MPI\_Accumulate).

Είναι σημαντικό να αναφέρουμε πως οι παραπάνω λειτουργίες επικοινωνίας αποτελούν μη παρεμποδιστικές διαδικασίες (non blocking). Έτσι, κατά την κλήση τους εκκινούν τη διαδικασία της απομακρυσμένης προσπέλασης και επιστρέφουν, χωρίς να αναμένουν την ολοκλήρωσή της.

### 1.3 Λειτουργίες συγχρονισμού

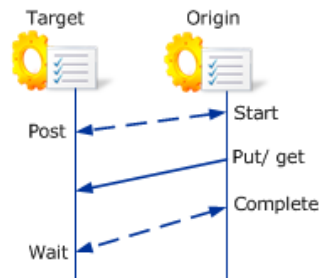
Σύμφωνα με τη σημασιολογία των μονόπλευρων επικοινωνιών, μόλις μια κλήση απομακρυσμένης προσπέλασης δεδομένων (όπως η MPI\_Put) επιστρέψει, η ολοκλήρωση της λειτουργίας της δεν είναι διασφαλισμένη. Για να είμαστε σίγουροι ότι μια λειτουργία απομακρυσμένης προσπέλασης δεδομένων ολοκληρώθηκε πρέπει να χρησιμοποιήσουμε διάφορες λειτουργίες συγχρονισμού. Οι λειτουργίες συγχρονισμού που παρέχει το MPI-2 για τις μονόπλευρες επικοινωνίες μπορούν να ταξινομηθούν ως ενεργές (active) και παθητικές (passive). Στις ενεργές λειτουργίες συγχρονισμού εμπλέκονται τόσο η διεργασία προέλευσης, όσο και η διεργασία στόχος. Αντίθετα, στις παθητικές λειτουργίες συγχρονισμού εμπλέκεται μόνο η διεργασία προέλευσης.



Σχήμα 1: Συγχρονισμός με χρήση φραχτών (fences)

Ο πρώτος μηχανισμός συγχρονισμού που παρέχει το MPI στηρίζεται στη συνάρτηση MPI\_Win\_fence. Η κλήση της MPI\_Win\_fence είναι συλλογική για όλη την ομάδα των διεργασιών. Η διεργασία προέλευσης χρησιμοποιεί αυτή τη συνάρτηση έτσι ώστε να εκκινήσει και να τερματίσει την περίοδο προσπέλασης. Από την άλλη μεριά, η διεργασία στόχος χρησιμοποιεί αυτή τη συνάρτηση ώστε να εκκινήσει και να τερματίσει την περίοδο έκθεσης. Όπως φαίνεται και στο σχήμα 1 η πρώτη κλήση της συνάρτησης MPI\_Win\_fence εκκινεί την περίοδο προσπέλασης ή έκθεσης, ενώ η αμέσως επόμενη κλήση αυτής της συνάρτησης την τερματίζει. Ένα μειονέκτημα αυτού του μηχανισμού για συγχρονισμό αποτελεί το γεγονός ότι η συνάρτηση MPI\_Win\_fence είναι συλλογική, οπότε αν ένα μικρό υποσύνολο

των διεργασιών της ομάδας επικοινωνούν μεταξύ τους τότε έχουμε ως αποτέλεσμα περιττό κόστος συγχρονισμού.



Σχήμα 2: Σύγχρονισμός μεταξύ ενός ζεύγους διεργασιών με χρήση των συναρτήσεων MPI\_Win\_post/ start/ complete/ wait

Για να ξεπεραστεί το κύριο μειονέκτημα του προηγούμενου μηχανισμού συγχρονισμού, το MPI ορίζει ένα δεύτερο μηχανισμό συγχρονισμού ο οποίος συγχρονίζει μόνο το υποσύνολο των διεργασιών που επικοινωνούν μεταξύ τους (σχήμα 2). Μια διεργασία που επιθυμεί να θέσει το παράθυρό της σε περίοδο έκθεσης καλεί τη συνάρτηση MPI\_Win\_post, η οποία λαμβάνει ένα όρισμα που καθορίζει την ομάδα των διεργασιών που μπορούν να προσπελάσουν το παράθυρό της. Μια διεργασία η οποία επιθυμεί να προσπελάσει ένα απομακρυσμένο παράθυρο μνήμης καλεί τη συνάρτηση MPI\_Win\_start στην οποία περνάει ως όρισμα την ομάδα των διεργασιών στόχων. Με τον τρόπο αυτό ξεκινάει η περίοδος προσπέλασης. Μόλις η διεργασία προέλευσης ολοκληρώσει την απομακρυσμένη προσπέλαση δεδομένων, καλεί τη συνάρτηση MPI\_Win\_complete έτσι ώστε να τερματίσει την περίοδο προσπέλασης. Η διεργασία στόχος καλεί τη συνάρτηση MPI\_Win\_wait για να τερματίσει την περίοδο έκθεσης.



Σχήμα 3: Σύγχρονισμός με χρήση αντικειμένων κλειδώματος

Οι δύο παραπάνω μέθοδοι συγχρονισμού που περιγράψαμε ανήκουν στην κατηγορία των ενεργητικών μεθόδων συγχρονισμού, διότι και τα δύο άκρα της επικοινωνίας συμμετέ-

χουν στον καθορισμό των παραμέτρων της επικοινωνίας. Ωστόσο, το MPI παρέχει και ένα τρίτο μηχανισμό ο οποίος είναι παθητικός, δηλαδή ο καθορισμός των παραμέτρων της επικοινωνίας πραγματοποιείται μόνο από τη διεργασία προέλευσης (σχήμα 3). Κατά τη χρήση αυτού του μηχανισμού, η διεργασία προέλευσης καλεί τη συνάρτηση `MPI_Win_lock` έτσι ώστε να αποκτήσει είτε αποκλειστική (*exclusive*), είτε κοινόχρηστη (*shared*) πρόσβαση στο παράθυρο της διεργασίας στόχου. Αφού ολοκληρώσει την προσπέλαση του απομακρυσμένου παραθύρου καλεί τη συνάρτηση `MPI_Win_unlock`. Μόλις η συνάρτηση `MPI_Win_unlock` επιστρέψει, το MPI διασφαλίζει ότι όλες οι προηγούμενες λειτουργίες προσπέλασης στο απομακρυσμένο παράθυρο μνήμης έχουν ολοκληρωθεί.

## 2 Χρήση των μονόπλευρων επικοινωνιών

Σε αυτή την ενότητα αναφέρουμε τις βασικές συναρτήσεις που παρέχει το MPI-2 για μονόπλευρες επικοινωνίες και περιγράφουμε σύντομα τον τρόπο με τον οποίο χρησιμοποιούνται.

### 2.1 Διαχείριση παραθύρων μνήμης

Για να επιτρέψει μια διεργασία την πρόσβαση άλλων διεργασιών σε μια περιοχή της μνήμης της, πρέπει να δημιουργήσει ένα παράθυρο μνήμης το οποίο θα την περιγράφει. Ένα παράθυρο μνήμης μπορεί να δημιουργηθεί με τη χρήση της συλλογικής συνάρτησης `MPI_Win_create`, η οποία επιστρέφει ένα δείκτη σε ένα αντικείμενο παραθύρου. Αυτό το αντικείμενο παραθύρου αντιπροσωπεύει την ομάδα των διεργασιών στον *communicator* που έχει καθοριστεί, καθώς και τα χαρακτηριστικά του κάθε παραθύρου. Η συνάρτηση δημιουργίας του παραθύρου μνήμης έχει την εξής μορφή:

```
i MPI_Win_create  
int MPI_Win_create(void *base, MPI_Aint size,  
    int disp_unit, MPI_Info info, MPI_Comm comm,  
    MPI_Win *win);
```

Οι παράμετροι της συνάρτησης αυτής είναι η αρχική διεύθυνση του παραθύρου (*base*), το μέγεθος του παραθύρου σε bytes (*size*), το μέγεθος μιας βασικής μονάδας του παραθύρου σε bytes (*disp\_unit*), ένα αντικείμενο που περιέχει πληροφορίες για το παράθυρο (*info*), ο *communicator* του *group* των διεργασιών (*comm*) και τέλος ένας δείκτης στο νέο αντικείμενο παραθύρου που επιστρέφεται (*win*).

Οι διεργασίες που ανήκουν στον *communicator* μπορούν να καθορίσουν διαφορετικές περιοχές μνήμης ως παράθυρα, διαφορετικά μεγέθη, καθώς και διαφορετικές ιδιότητες για

το παράθυρό τους. Ωστόσο, είναι αναγκαίο όλες οι διεργασίες που ανήκουν στο group να καλέσουν τη συνάρτηση δημιουργίας παραθύρου. Μια διεργασία μπορεί να θέσει το πεδίο size της κλήσης ίσο με μηδέν έτσι ώστε να μην εκθέσει κάποια περιοχή της μνήμης της στις υπόλοιπες.

Οι πληροφορίες ενός αντικειμένου παραθύρου αποθηκεύονται προσωρινά και μπορούν να ανακτηθούν με χρήση της συνάρτησης `MPI_Win_get_attr`. Η συνάρτηση αυτή ορίζεται ως εξής:

```
i MPI_Win_get_attr  
| int MPI_Win_get_attr(MPI_Win win, int keyVal,  
| void *attrVal, int *flag);
```

Η μεταβλητή `win` αναφέρεται στο αντικείμενο παραθύρου για το οποίο επιθυμούμε να ανακτήσουμε κάποια ιδιότητά του, ενώ η παράμετρος `keyVal` ταυτοποιεί την ιδιότητα την οποία θέλουμε να ανακτήσουμε. Για παράδειγμα, αν περάσουμε ως όρισμα τη σταθερά `MPI_WIN_BASE` τότε ο δείκτης `attrVal` μετά την κλήση της συνάρτησης θα δείχνει στην αρχή του παραθύρου `win`. Αντίστοιχα, χρησιμοποιώντας τη σταθερά `MPI_WIN_SIZE`, η θέση στην οποία δείχνει ο δείκτης `attrVal` θα περιέχει το μέγεθος του παραθύρου σε bytes. Τέλος, αν χρησιμοποιήσουμε τη σταθερά `MPI_WIN_DISP_UNIT`, τότε η θέση στην οποία δείχνει ο δείκτης `attrVal` θα περιέχει το μέγεθος της βασικής μονάδας του παραθύρου σε bytes.

Από την άλλη μεριά, η διαγραφή ενός παραθύρου πραγματοποιείται με τη συνάρτηση `MPI_Win_free`. Η συνάρτηση αυτή ορίζεται ως εξής:

```
i MPI_Win_free  
| int MPI_Win_free(MPI_Win *win);
```

Ο δείκτης `win` στην παραπάνω συνάρτηση αναφέρεται στο παράθυρο μνήμης που επιθυμούμε να διαγράψουμε. Μόλις η συνάρτηση αυτή επιστρέψει, ο δείκτης αυτός θα έχει τεθεί στην τιμή `MPI_WIN_NULL`. Η συνάρτηση `MPI_Win_free` αποτελεί μια συλλογική λειτουργία και πρέπει να κληθεί από όλες της διεργασίες του group.

Ένα παράδειγμα χρήσης των συναρτήσεων δημιουργίας και διαγραφής παραθύρων φαίνεται στη λίστα 1. Σε αυτό το παράδειγμα, η διεργασία 0 δημιουργεί ένα παράθυρο μνήμης ώστε να παρέχει πρόσβαση στις υπόλοιπες διεργασίες του group στον χώρο μνήμης που καταλαμβάνει ο πίνακας `array`. Οι υπόλοιπες διεργασίες δεν εκθέτουν κάποια περιοχή της μνήμης τους, ωστόσο καλούν και αυτές τη συνάρτηση δημιουργίας παραθύρου. Στη συνάρτηση αυτή περνούν ως όρισμα το `MPI_BOTTOM` στη διεύθυνση βάσης και 0 στο μέγεθος του παραθύρου.

---

Λίστα 1: Παράδειγμα δημιουργίας και διαγραφής παραθύρου μνήμης

---

```
double array[1000];
int rank;
MPI_Win window;
MPI_Aint win_size;

MPI_Comm_rank(MPI_COMM_WORLD, &rank);
If (rank == 0) {
    win_size = 1000 * sizeof(double);
    MPI_Win_create(array, win_size, sizeof(double),
                   MPI_INFO_NULL, MPI_COMM_WORLD, &window);
} else {
    MPI_Win_create(MPI_BOTTOM, 0, 1, MPI_INFO_NULL,
                   MPI_COMM_WORLD, &window);
}


MPI_Win_free(window);
```

---

## 2.2 Απομακρυσμένη προσπέλαση δεδομένων

Το πρότυπο MPI-2 παρέχει τρεις συναρτήσεις που μπορούν να χρησιμοποιηθούν για απομακρυσμένη προσπέλαση δεδομένων. Οι συναρτήσεις αυτές είναι μη παρεμποδιστικές. Έτσι, κατά την κλήση τους εκκινούν τη λειτουργία απομακρυσμένης προσπέλασης και έπειτα επιστρέφουν, χωρίς να αναμένουν την ολοκλήρωσή της.

Η συνάρτηση που παρέχει το MPI-2 για απομακρυσμένη εγγραφή είναι η MPI\_Put. Η συνάρτηση αυτή ορίζεται ως εξής:

 **MPI\_Put**

```
int MPI_Put(void *origAddress, int origCount,
            MPI_Datatype origType, int targetRank,
            MPI_Aint targetDisp, int targetCount,
            MPI_Datatype targetType, MPI_Win win);
```

Ας περιγράψουμε τα ορίσματα της παραπάνω συνάρτησης. Το όρισμα origAddress είναι ένας δείκτης στην περιοχή μνήμης που είναι τοποθετημένα τα δεδομένα που θέλουμε να αντιγραφούν στο απομακρυσμένο παράθυρο μνήμης. Η παράμετρος origCount είναι ο αριθμός των στοιχείων που περιέχει η παραπάνω περιοχή μνήμης, ενώ το όρισμα origType

είναι ο τύπος των στοιχείων αυτών. Το όρισμα `targetRank` είναι η ταυτότητα της απομακρυσμένης διεργασίας στόχου. Το παράθυρο μνήμης που επιθυμούμε να προσπελάσουμε περιγράφεται από την παράμετρο `win` και περιλαμβάνει `targetCount` στοιχεία τύπου `targetType`. Τέλος, θέτοντας στην παράμετρο `targetDisp` μια θετική τιμή διάφορη του μηδενός μπορούμε να αναφερθούμε σε μια συγκεκριμένη θέση μνήμης στο εσωτερικό του παραθύρου μνήμης της διεργασίας στόχου. Μπορούμε να φανταστούμε αυτή την παράμετρο σαν ένα `offset` από την αρχική διεύθυνση μνήμης που περιλαμβάνει το παράθυρο της διεργασίας στόχου.

Για την απομακρυσμένη ανάγνωση δεδομένων από μια διεργασία το MPI-2 παρέχει τη συνάρτηση `MPI_Get`. Η συνάρτηση αυτή ορίζεται ως εξής:

```
i MPI_Get
int MPI_Get(void *origAddress, int origCount,
            MPI_Datatype origType, int targetRank,
            MPI_Aint targetDisp, int targetCount,
            MPI_Datatype targetType, MPI_Win win);
```

Τα ορίσματα της συνάρτησης αυτής είναι παρόμοια με εκείνα της `MPI_Put`. Η μόνη διαφορά είναι ότι η μεταφορά δεδομένων γίνεται από το απομακρυσμένο παράθυρο μνήμης προς την περιοχή `origAddress`.

Η τρίτη συνάρτηση που παρέχει το MPI-2 για απομακρυσμένη προσπέλαση δεδομένων είναι η `MPI_Accumulate`. Η συνάρτηση αυτή ορίζεται ως εξής:

```
i MPI_Accumulate
int MPI_Accumulate t(void *origAddress, int origCount,
                    MPI_Datatype origType, int targetRank,
                    MPI_Aint targetDisp, int targetCount,
                    MPI_Datatype targetType, MPI_Op op, MPI_Win win);
```

Τα ορίσματα της συνάρτησης αυτής είναι παρόμοια με εκείνα της `MPI_Put`. Ωστόσο, επειδή η συνάρτηση `MPI_Accumulate` δεν επανεγγράφει τα παλιά δεδομένα του απομακρυσμένου παραθύρου, αλλά συνδυάζει τα παλιά με τα νέα, δέχεται ένα επιπλέον όρισμα (`op`), το οποίο αντιστοιχεί στην πράξη η οποία θα εφαρμοστεί κατά τον συνδυασμό των παλαιών με των νέων δεδομένων. Η τιμή αυτού του ορίσματος μπορεί να είναι οποιαδήποτε από τις `MPI_MAX/ MIN/ SUM/ PROD/ LAND/ BAND/ LOR/ BOR/ LXOR/ BXOR`.

## 2.3 Συγχρονισμός διεργασιών

Όπως αναφέραμε, για να είμαστε σίγουροι ότι μια λειτουργία απομακρυσμένης προσπέλασης δεδομένων ολοκληρώθηκε πρέπει να χρησιμοποιήσουμε διάφορες λειτουργίες συγχρονισμού. Η πρώτη μέθοδος συγχρονισμού που αναφέραμε στηρίζεται στη χρήση της



## Λίστα 2: Παράδειγμα χρήσης της MPI\_Win\_fence

---

```
int Array[SIZE];
MPI_Group comm_group, group;
MPI_Win win;
int rank;
...
MPI_Comm_rank(MPI_COMM_WORLD, &rank);
if (rank == 0) {
    MPI_Win_create(Array, SIZE * sizeof(int), sizeof(int),
                   MPI_INFO_NULL, MPI_COMM_WORLD, &win);
    MPI_Win_fence(0, win);
    MPI_Win_fence(0, win);
    print_results(Array);
} else if (rank == 1) {
    MPI_Win_create(NULL, 0, 1, MPI_INFO_NULL,
                   MPI_COMM_WORLD, &win);
    MPI_Win_fence(0, win);
    for (i = 0; i < SIZE; i++)
        MPI_Put(&i, 1, MPI_INT, 1, i, 1, MPI_INT, win);
    MPI_Win_fence(&win);
}
...
```

---

συνάρτησης MPI\_Win\_fence. Η συνάρτηση αυτή είναι συλλογική και καλείται από όλες τις διεργασίες της ομάδας που συσχετίζεται με ένα παράθυρο μνήμης. Το πρότυπο της συνάρτησης αυτής είναι το εξής:



### **MPI\_Win\_fence**


```
int MPI_Win_fence(int assert, MPI_Win win);
```

Το όρισμα assert επιτρέπει τον καθορισμό επιπρόσθετων πληροφοριών για τη λειτουργία του συγχρονισμού. Δυνατές τιμές είναι οι: 0, MPI\_MODE\_NOSTORE/ NOPUT/ NOPRECEDE/ NOSUCCEED. Η παράμετρος win καθορίζει το παράθυρο για το οποίο επιθυμούμε να γίνει ο συγχρονισμός.

Είναι σημαντικό να τονίσουμε ότι η συνάρτηση MPI\_Win\_fence πρέπει πάντα να καλείται δύο φορές, μια πριν την έναρξη λειτουργιών απομακρυσμένης προσπέλασης και μια

μετά τον τερματισμό τους. Ένα παράδειγμα χρήσης της MPI\_Win\_fence διακρίνεται στη λίστα 2. Στο παράδειγμα αυτό, η διεργασία 0 δημιουργεί ένα παράθυρο μνήμης και το εκθέτει καλώντας την MPI\_Win\_fence. Από την άλλη μεριά, η διεργασία 1 χρησιμοποιώντας την MPI\_Win\_fence ξεκινάει την περίοδο προσπέλασής της και τοποθετεί κάποια δεδομένα στο απομακρυσμένο παράθυρο. Έπειτα, καλεί ξανά την MPI\_Win\_fence ώστε να τερματίσει την περίοδο προσπέλασης. Κάτι αντίστοιχο κάνει και η διεργασία 0 τερματίζοντας την περίοδο έκθεσης. Στο σημείο αυτό, τα δεδομένα που έχει τοποθετήσει η διεργασία 1 στη μνήμη της διεργασίας 0 είναι διαθέσιμα και η διεργασία 0 μπορεί να τα προσπελάσει.

Η δεύτερη μέθοδος συγχρονισμού που παρέχει το MPI-2 στηρίζεται στη χρήση τεσσάρων συναρτήσεων: MPI\_Win\_post, MPI\_Win\_start, MPI\_Win\_complete και MPI\_Win\_wait. Η πρώτη από τις παραπάνω συναρτήσεις καλείται από τη διεργασία στόχο ώστε να θέσει το παράθυρό της σε περίοδο έκθεσης. Η δεύτερη, καλείται από τη διεργασία προέλευσης ώστε να δηλώσει την αρχή της περιόδου προσπέλασης. Ομοίως, η τρίτη συνάρτηση καλείται από τη διεργασία προέλευσης ώστε να δηλώσει τον τερματισμό της περιόδου προσπέλασης. Τέλος, η τελευταία συνάρτηση καλείται από τη διεργασία στόχο ώστε να δηλώσει τον τερματισμό της περιόδου έκθεσης. Οι συναρτήσεις αυτές ορίζονται ως εξής:

 **MPI\_Win\_post/ start/ complete/ wait**

```
int MPI_Win_post(MPI_Group group, int assert, MPI_Win win);  
int MPI_Win_start(MPI_Group group, int assert, MPI_Win win);  
int MPI_Win_complete(MPI_Win win);  
int MPI_Win_wait(MPI_Win win);
```

Στις παραπάνω συναρτήσεις, το όρισμα assert επιτρέπει τον καθορισμό επιπρόσθετων πληροφοριών για τη λειτουργία του συγχρονισμού και είναι το ίδιο με το όρισμα που δέχεται η συνάρτηση MPI\_Win\_fence. Το όρισμα group καθορίζει την ομάδα των διεργασιών που θέλουμε να συγχρονιστούν. Όταν μια διεργασία στόχος εκθέσει το παράθυρό της καλώντας την MPI\_Win\_post, δίνει τη δυνατότητα σε όλες τις διεργασίες που ανήκουν στην ομάδα που καθορίζεται από την παράμετρο group να το προσπελάσουν. Αντίστοιχα, όταν μια διεργασία προέλευσης εκκινήσει μια περίοδο προσπέλασης καλώντας την MPI\_Win\_start, πρέπει όλες οι διεργασίες που ανήκουν στην ομάδα group να καλέσουν τη συνάρτηση MPI\_Win\_post. Τέλος, η παράμετρος win καθορίζει το παράθυρο για το οποίο επιθυμούμε να γίνει ο συγχρονισμός.

Ένα παράδειγμα χρήσης των MPI\_Win\_post/ start/ complete/ wait διακρίνεται στη λίστα 3. Στο παράδειγμα αυτό, η διεργασία 0 δημιουργεί ένα παράθυρο μνήμης και το εκθέτει καλώντας την MPI\_Win\_post. Από την άλλη μεριά, η διεργασία 1 χρησιμοποιώντας την MPI\_Win\_start ξεκινάει την περίοδο προσπέλασής της και τοποθετεί κάποια δεδομένα στο απομακρυσμένο παράθυρο. Έπειτα, καλεί την MPI\_Win\_complete ώστε να τερματίσει την

Λίστα 3: Παράδειγμα χρήσης των MPI\_Win\_post/ start/ complete/ wait

---

```
int Array[SIZE];
MPI_Group comm_group, group;
MPI_Win win;
int rank, destrank;
...
MPI_Comm_rank(MPI_COMM_WORLD, &rank);
MPI_Comm_group(MPI_COMM_WORLD, &comm_group);
if (rank == 0) {
    MPI_Win_create(Array, SIZE * sizeof(int), sizeof(int),
                   MPI_INFO_NULL, MPI_COMM_WORLD, &win);

    destrank = 0;
    MPI_Group_incl(comm_group, 1, &destrank, &group);
    MPI_Win_post(group, 0, win);
    MPI_Win_wait(win);
    print_results(Array);
} else if (rank == 1) {
    MPI_Win_create(NULL, 0, 1, MPI_INFO_NULL,
                   MPI_COMM_WORLD, &win);

    destrank = 1;
    MPI_Group_incl(comm_group, 1, &destrank, &group);
    MPI_Win_start(group, 0, win);
    for (i = 0; i < SIZE; i++)
        MPI_Put(&i, 1, MPI_INT, 1, i, 1, MPI_INT, win);
    MPI_Win_complete(win);
}
...
```

---

περίοδο προσπέλασης. Κάτι αντίστοιχο κάνει και η διεργασία 0 καλώντας τη συνάρτηση `MPI_Win_wait` τερματίζοντας την περίοδο έκθεσης. Στο σημείο αυτό, τα δεδομένα που έχει τοποθετήσει η διεργασία 1 στη μνήμη της διεργασίας 0 είναι διαθέσιμα και η διεργασία 0 μπορεί να τα προσπελάσει. Είναι σημαντικό να τονίσουμε ότι η `MPI_Win_wait` είναι μια παρεμποδιστική μέθοδος και η κλήση της ολοκληρώνεται μόλις η διεργασία προέλευσης καλέσει την `MPI_Win_complete`.

Το τελευταίο είδος συγχρονισμού που παρέχει το MPI-2 στηρίζεται στη χρήση αμοιβαίου αποκλεισμού. Οι συναρτήσεις συγχρονισμού που παρέχονται για αυτό το μοντέλο είναι οι `MPI_Win_lock`, `MPI_Win_unlock` και καλούνται μόνο από τη διεργασία προέλευσης. Να τονίσουμε ότι σε αυτό το μοντέλο η διεργασία στόχος δε συμμετέχει στη διαδικασία του συγχρονισμού. Οι συναρτήσεις αυτές έχουν την εξής μορφή:

```
❗ MPI_Win_lock/ unlock  
int MPI_Win_lock(int lockType, int rank, int assert,  
                 MPI_Win win);  
int MPI_Win_unlock(int rank, MPI_Win win);
```

Η παράμετρος `lockType` μπορεί να πάρει είτε την τιμή `MPI_LOCK_EXCLUSIVE`, είτε την τιμή `MPI_LOCK_SHARED`. Στην πρώτη περίπτωση η πρόσβαση στο παράθυρο μνήμης της διεργασίας στόχου είναι αποκλειστική, ενώ στη δεύτερη περίπτωση είναι κοινόχρηστη. Η παράμετρος `rank` καθορίζει τη διεργασία στόχο της οποίας το παράθυρο μνήμης πρόκειται να προσπελαστεί, ενώ η παράμετρος `win` καθορίζει το παράθυρο. Τέλος, η παράμετρος `assert` επιτρέπει τον καθορισμό επιπρόσθετων πληροφοριών για τη λειτουργία του συγχρονισμού και είναι το ίδιο με το όρισμα που δέχεται η συνάρτηση `MPI_Win_fence`.

Ένα παράδειγμα χρήσης των `MPI_Win_lock/ unlock` διακρίνεται στη λίστα 4. Στο παράδειγμα αυτό, η διεργασία 0 δημιουργεί ένα παράθυρο μνήμης. Από την άλλη μεριά, οι υπόλοιπες διεργασίες χρησιμοποιώντας την `MPI_Win_lock` κλειδώνουν το παράθυρο και τοποθετούν σε αυτό κάποια δεδομένα. Επειδή στην `MPI_Win_lock` χρησιμοποιείται η παράμετρος `MPI_LOCK_EXCLUSIVE`, μόνο μια διεργασία θα καταφέρει να αποκτήσει την κλειδαριά, ώστε να προσπελάσει το παράθυρο. Οι υπόλοιπες θα μπλοκάρουν στην `MPI_Win_lock` μέχρι η διεργασία που έχει την κλειδαριά να την ελευθερώσει καλώντας την `MPI_Win_unlock`. Η διεργασία 0 δε συμμετέχει σε αυτή την επικοινωνία. Μόλις όλες οι διεργασίες τελειώσουν με την προσπέλαση του παραθύρου, η διεργασία 0 μπορεί να χρησιμοποιήσει τα δεδομένα του. Είναι φανερό πως αυτό το μοντέλο συγχρονισμού μοιάζει με το μοντέλο κοινόχρηστης μνήμης.

Λίστα 4: Παράδειγμα χρήσης των MPI\_Win\_lock/ unlock

---

```
int Array[SIZE];
MPI_Group comm_group, group;
MPI_Win win;
int rank;
...
MPI_Comm_rank(MPI_COMM_WORLD, &rank);
if (rank == 0) {
    MPI_Win_create(Array, SIZE * sizeof(int), sizeof(int),
        MPI_INFO_NULL, MPI_COMM_WORLD, &win);
} else if (rank == 1) {
    MPI_Win_create(NULL, 0, 1, MPI_INFO_NULL,
        MPI_COMM_WORLD, &win);
    MPI_Win_lock(MPI_LOCK_EXCLUSIVE, 0, 0, win);
    for (i = 0; i < SIZE; i++)
        MPI_Put(&i, 1, MPI_INT, 1, i, 1, MPI_INT, win);
    MPI_Win_unlock(0, win);
}
MPI_Barrier();
if (rank == 0) print_results(Array);
...

```

---