Επιμερισμός δεδομένων
Διαίρει και βασίλευε

Κ.Γ. Μαργαρίτης

προσαρμογή από το μάθημα του Barry Wilkinson
ITCS 4145/5145 2006Cluster Computing
Univ. of North Carolina at Charlotte

Επιμερισμός δεδομένων

Κατανομή δεδομένων (και εργασιών) σε επι μέρους τμήματα (άρα διεργασίες – και επεξεργαστές).

Διαίρει και βασίλευε

Επαναληπτική υποδιαίρεση του προβλήματος (δεδομένα ή εργασίες) σε υπο-προβλήματα, μέχρι να φθάσουμε σε πολύ απλές περιπτώσεις.

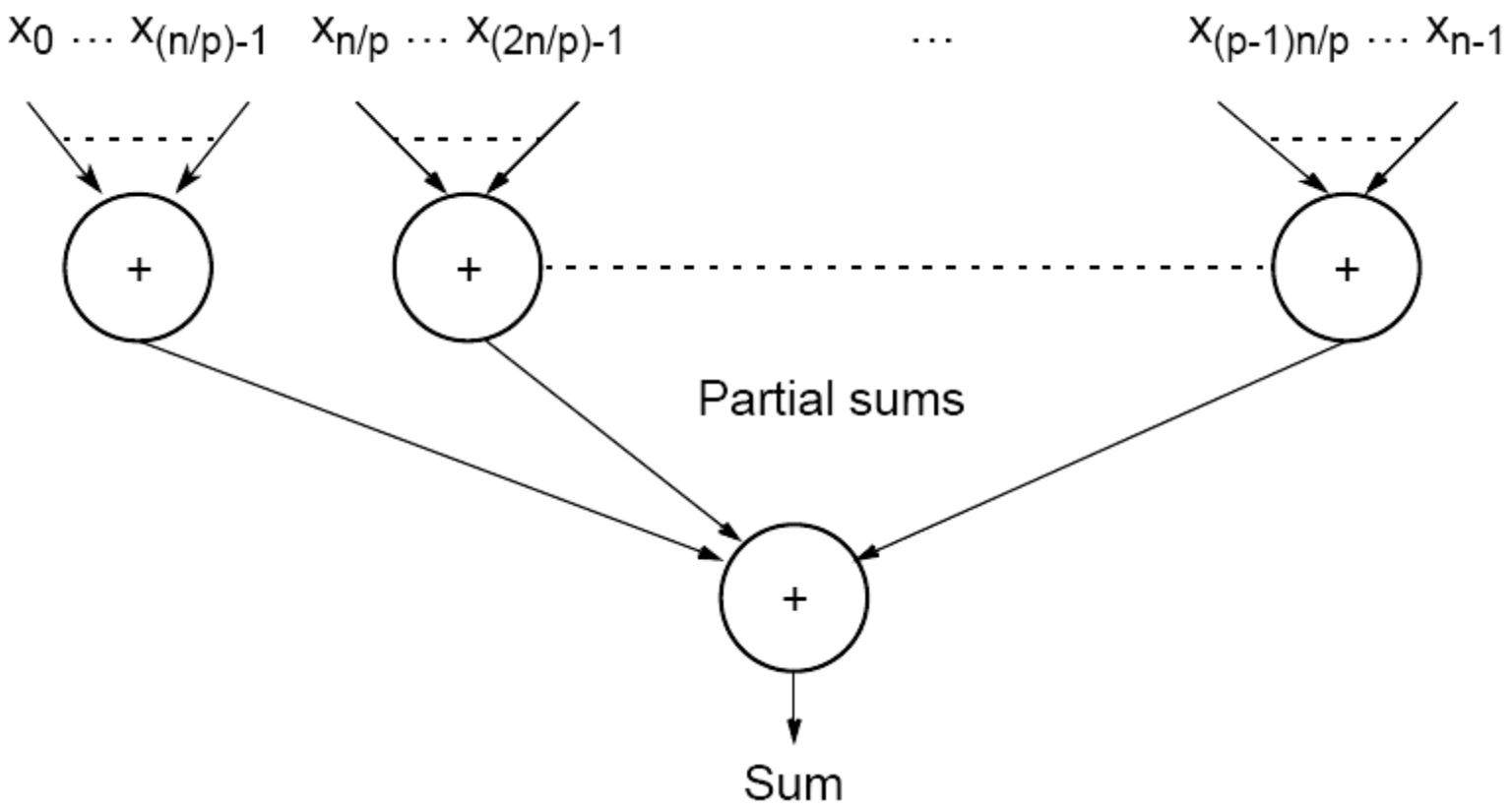
Αναδρομικές διαδικασίες μπορούν να εφαρμοστούν αν όμως εξασφαλιστεί η πρόσβαση των αναδρομικών διεργαισιών σε τοπικά δεδομένα.

Παραδείγματα

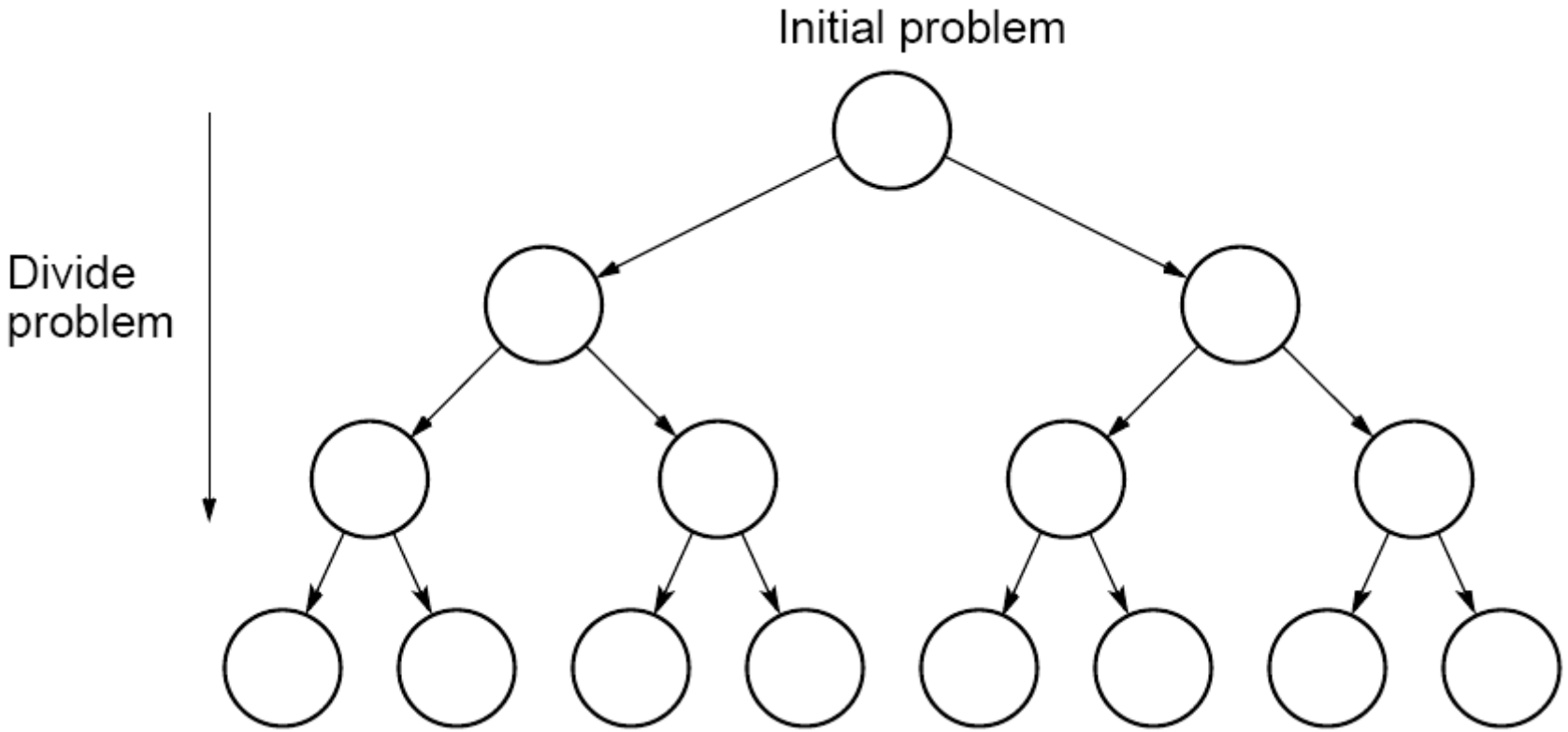
Πολλές περιπτώσεις:

- Πράξεις σε αριθμητικά – συμβολικά ανύσματα
- Αλγόριθμοι ταξινόμησης - αναζήτησης
- Αριθμητική ολοκλήρωση
- Πρόβλημα *N*-σωμάτων

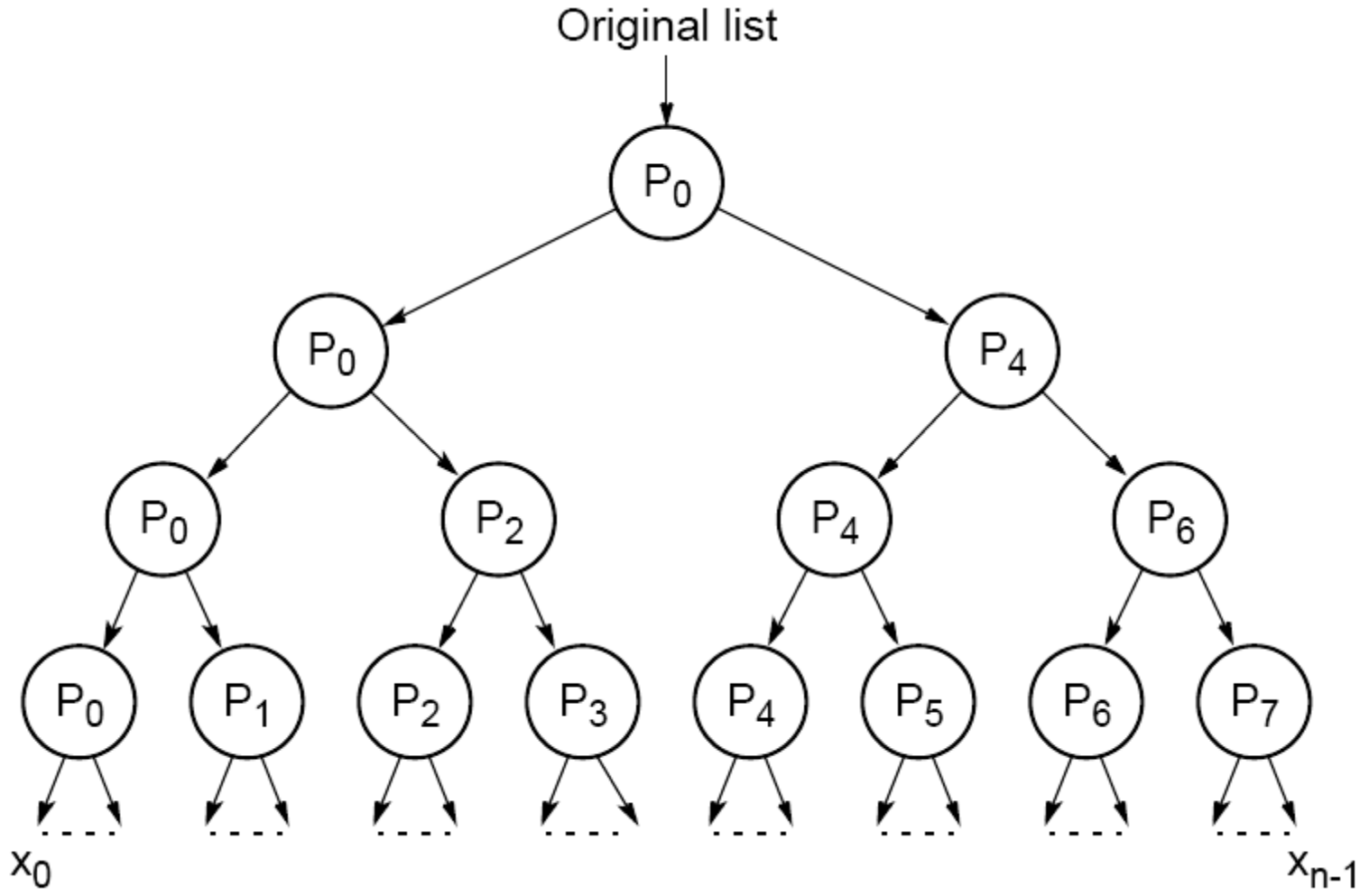
Άθροιση n αριθμών



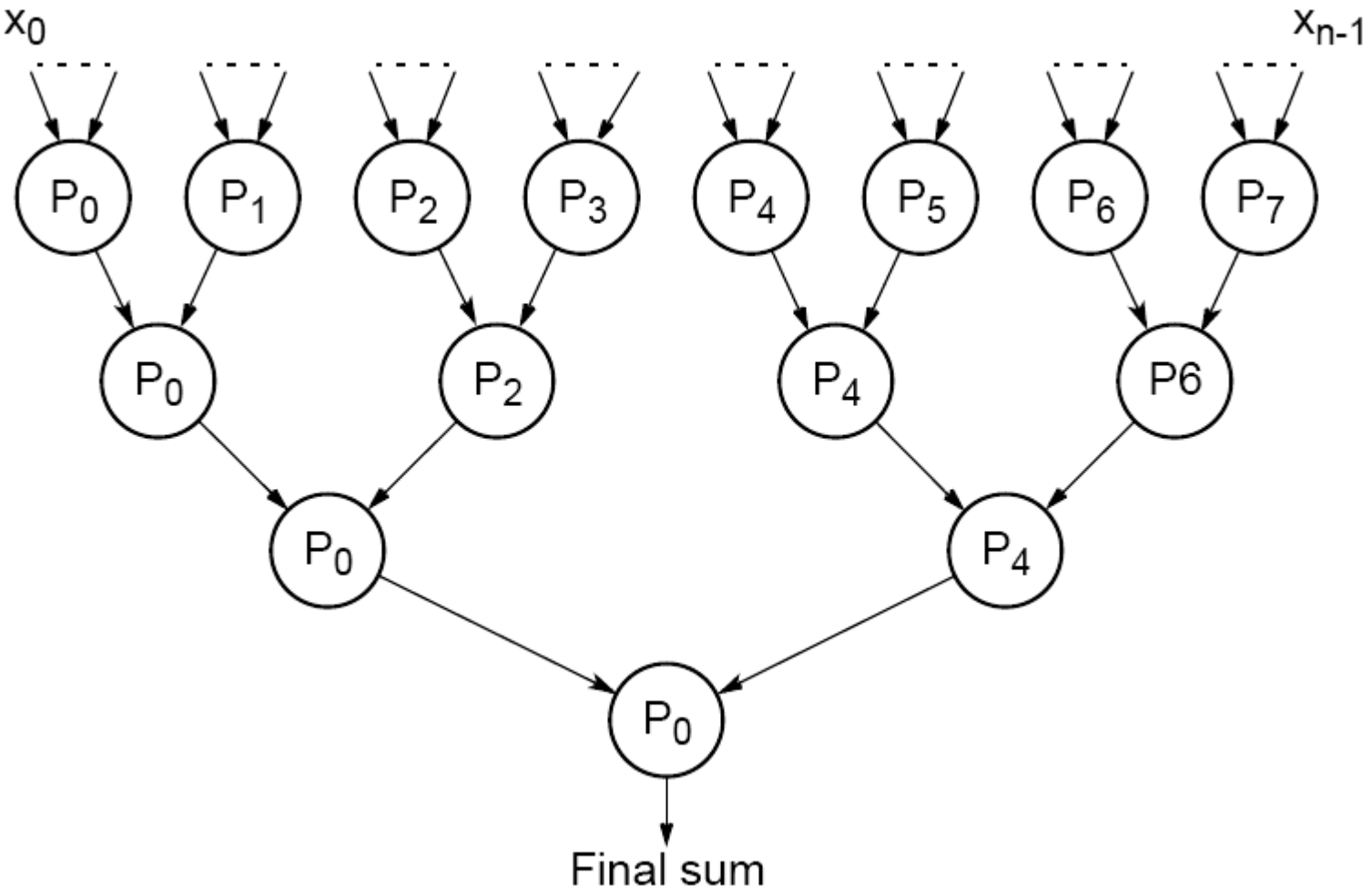
Κατασκευή δένδρου



Διαίρεση λίστας σε υπο-λίστες

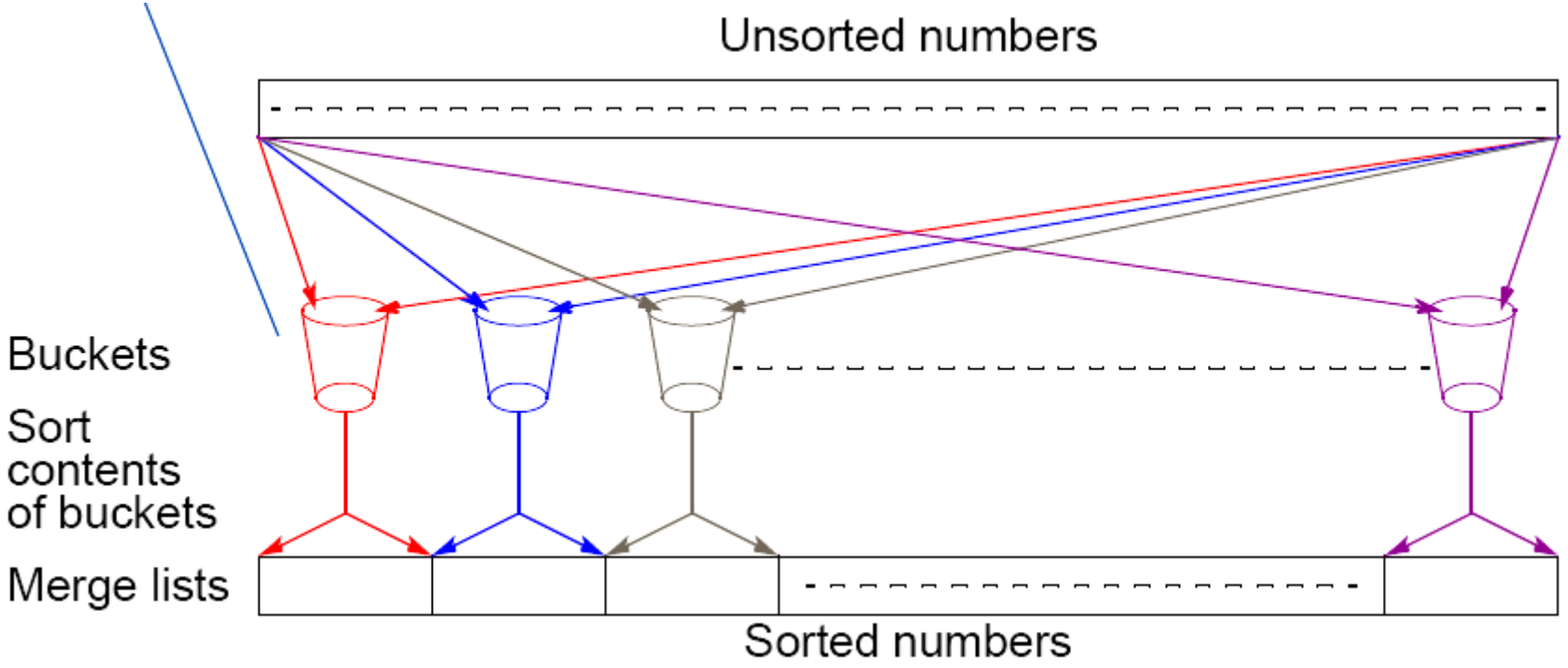


Μερικά αθροίσματα



Ταξινόμηση δοχείου (Bucket sort)

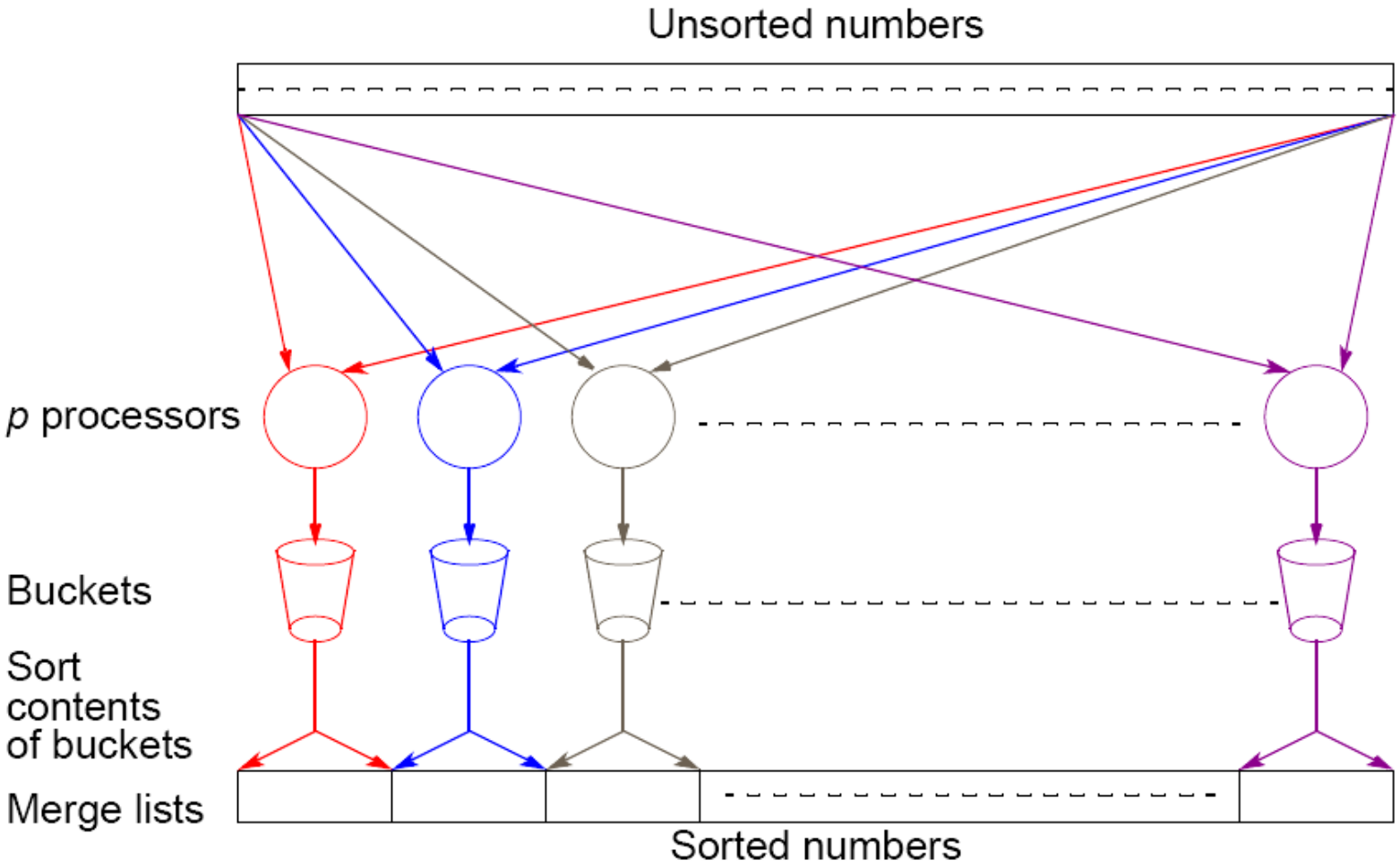
Κάθε “δοχείο” συλλέγει αριθμούς σε κάποιο εύρος.
Μέσα σε καθε “δοχείο” η ταξινόμηση είναι ακολουθιακή..



Πολυπλοκότητα ακολουθιακής ταξινόμησης: O(*n*log(*n*/*m*).
Δουλεύει καλά αν οι αταξινόμητοι αριθμοί είναι ομοιόμορφα κατανεμημένοι, ας πούμε στο διάστημα 0 to *a* - 1.

Απλή παραλληλοποίηση

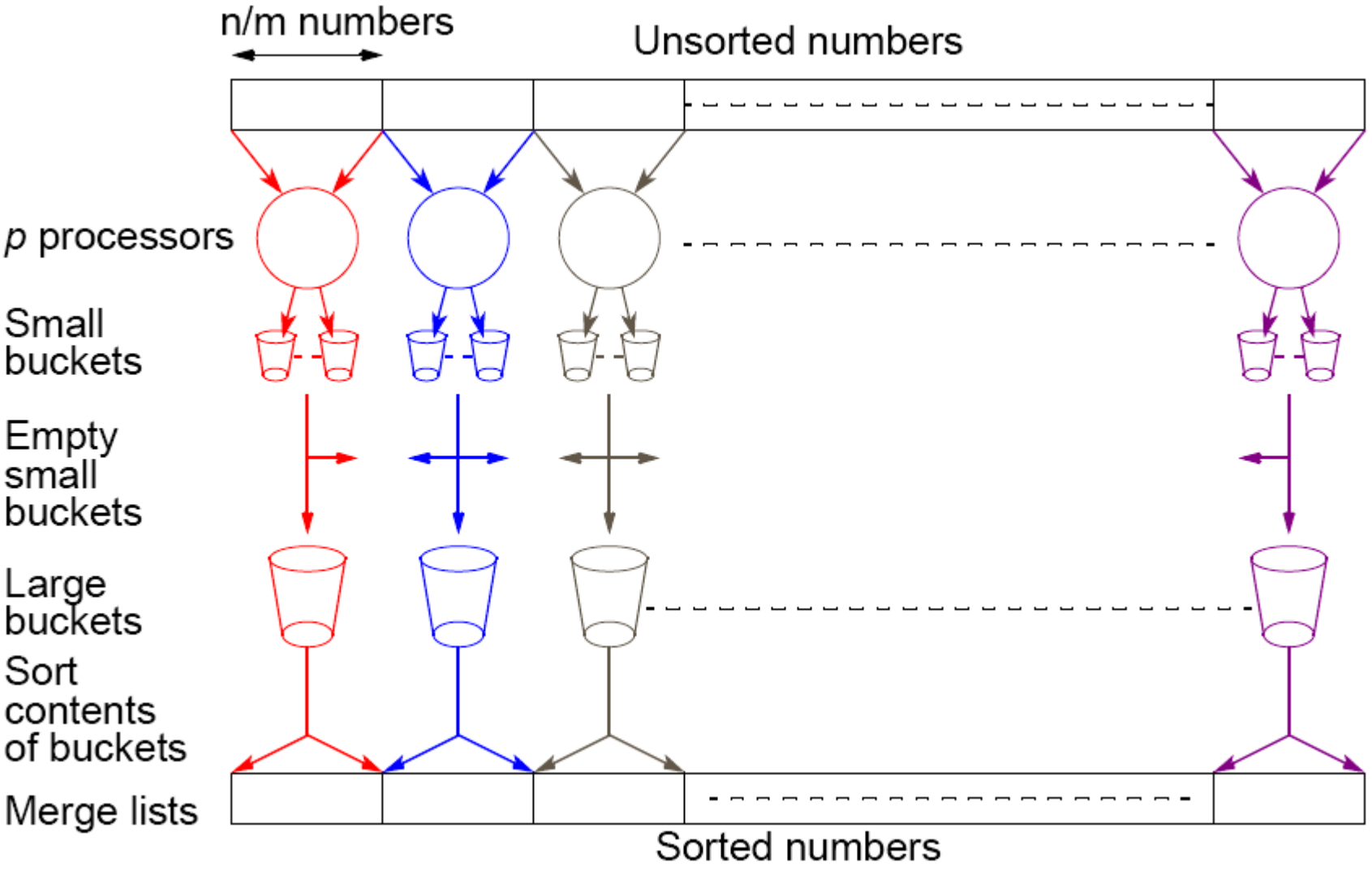
Ένα δοχείο για κάθε επεξεργαστή.



Περισσότερος παραλληλισμός

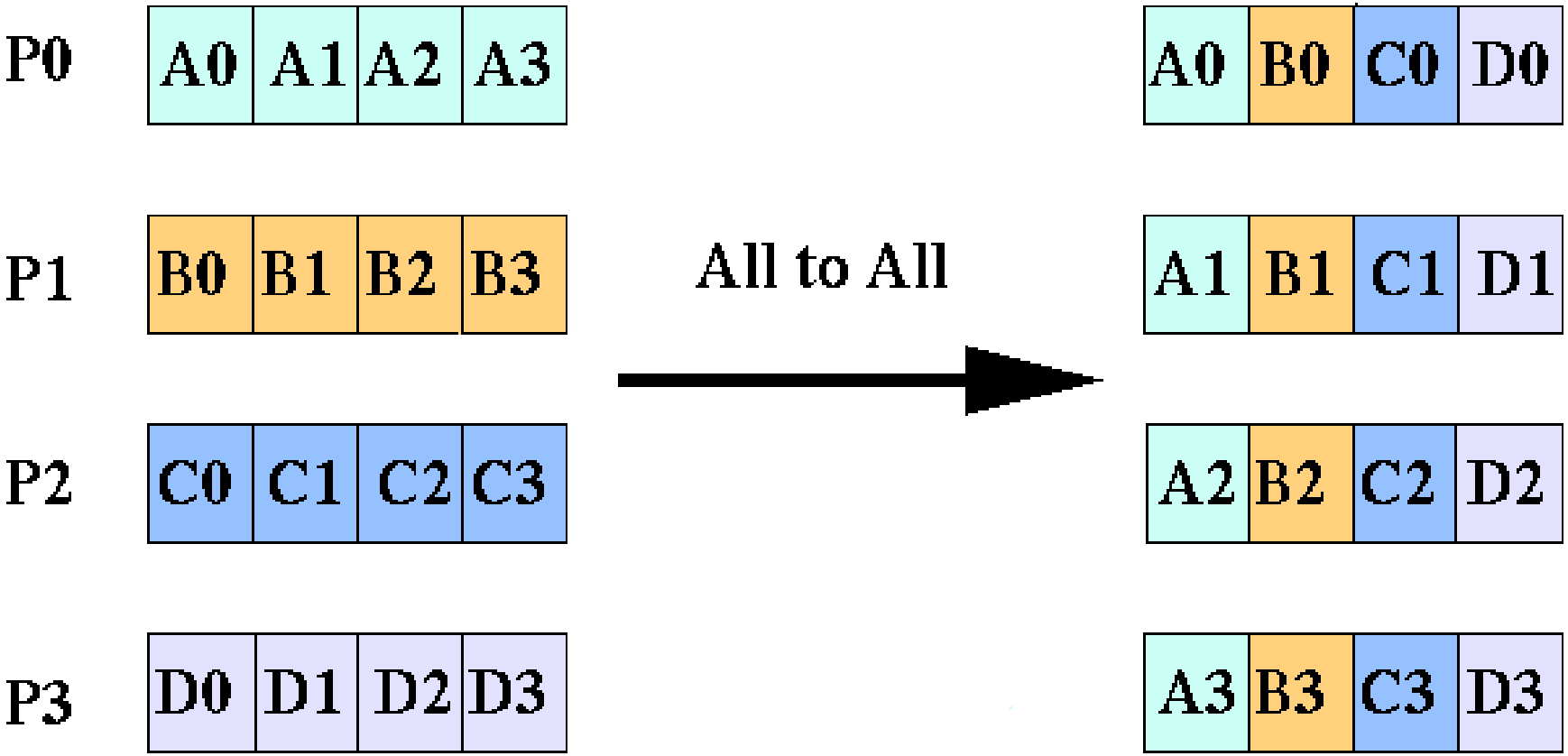
- Η αταξινόμητη ακολουθία διαιρείται σε p τμήματα.
- Κάθε επεξεργαστής ελέγχει ένα τμήμα.
- Κάθε επεξεργαστής έχει 1 “μεγάλο” δοχείο, για τη περιοχή του.
- Κάθε επεξεργαστής έχει p “μικρά” δοχεία, ένα για κάθε τμήμα.
- Κάθε επεξεργαστής τοποθετεί τους αριθμούς του τμήματός του στα “μικρά” δοχεία του.
- Το i (i=1,..p) “μικρό” δοχείο κάθε επεξεργαστή αδειάζει στο “μεγάλο” δοχείο του i (i=1, ..p) επεξεργαστή.
- Κάθε “μεγάλο” δοχείο ταξινομείται στον επεξεργαστή του.
- Τα ταξινομημένα δοχεία συντίθενται με τη σειρά του επεξεργαστή.

Περισσότερος παραλληλισμός (συν.)



all-to-all broadcast

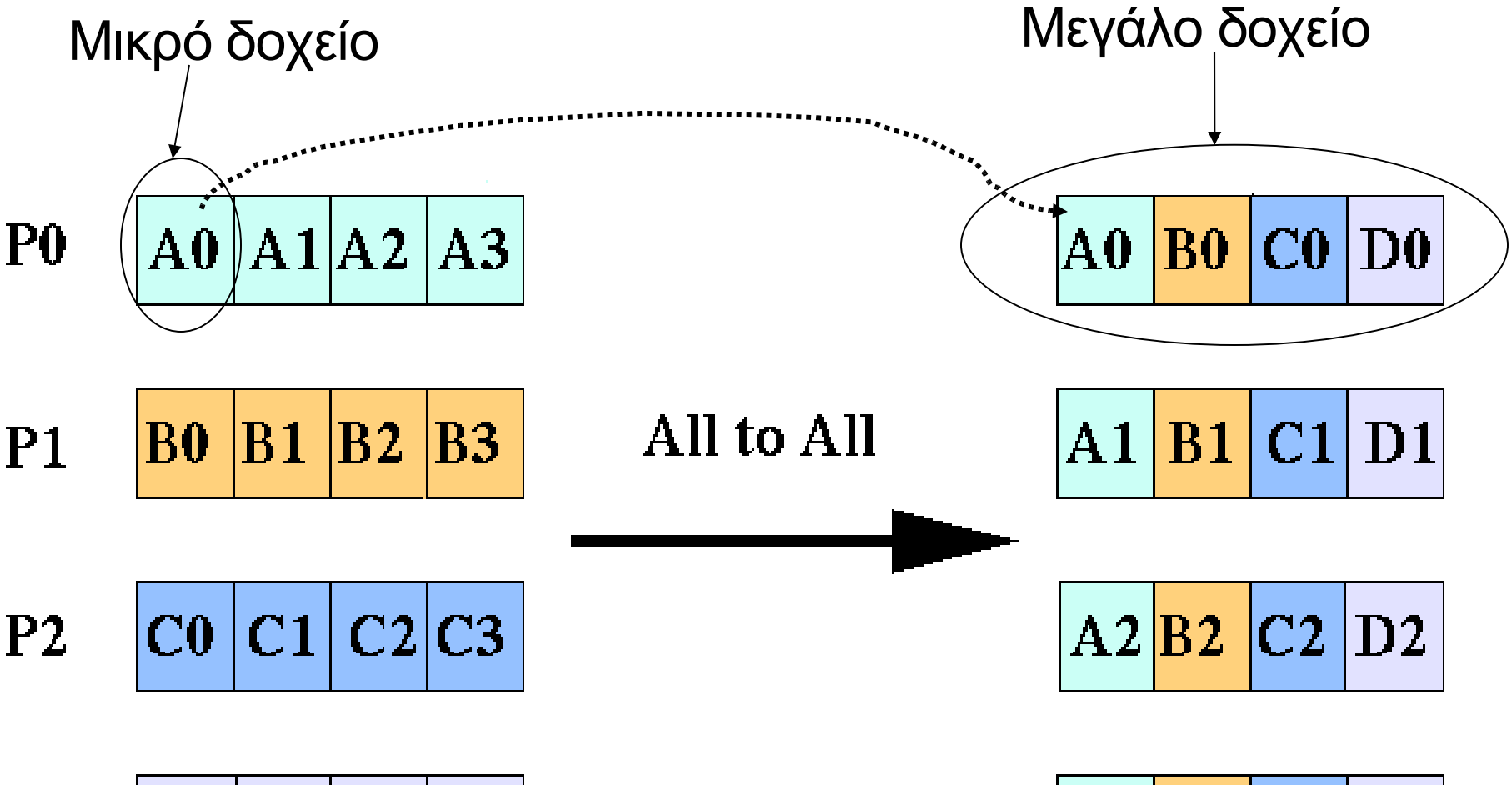
- Μια συνάρτηση MPI που μπορεί να εφαρμοστεί σ' αυτή τη περίπτωση.
- Στέλνει ένα διαφοτερικό στοιχείο από κάθε διεργασία προς κάθε άλλη διεργασία.
- Αντιστοιχεί σε πολλαπλές συναρτήσεις διανομής (scatter).
- Καλύτερα all-to-all scatter?



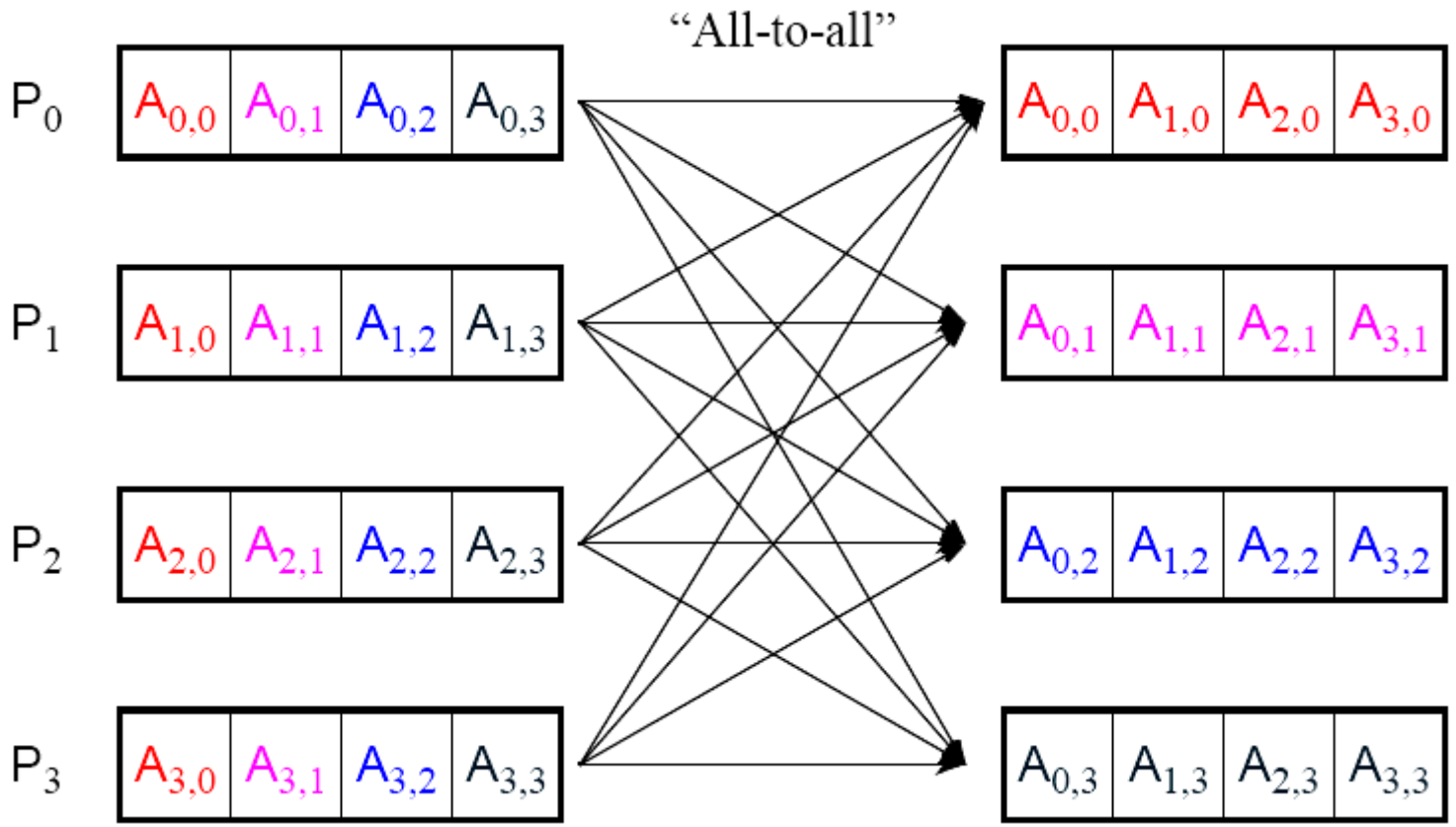
From: http://www.mhpcc.edu/training/workshop/parallel_intro/MAIN.html

Εφαρμογή all-to-all broadcast για το άδεισμα μικρών δοχείων στα μεγάλα

Έστω 4 τμήματα και 4 επεξεργαστές



Το all-to-all broadcast μεταφέρει γραμμές πίνακα σε στήλες: Αναστροφή πίνακα.

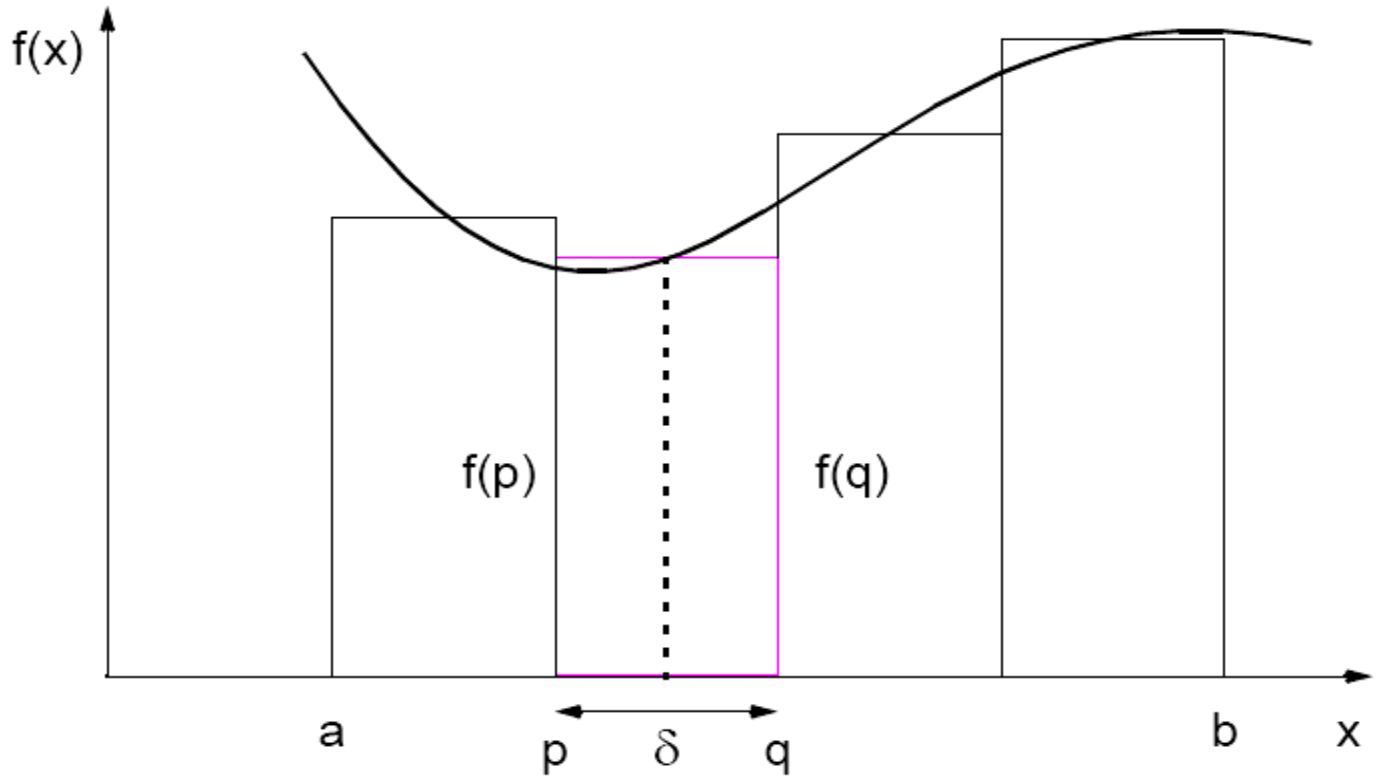


Αριθμητική ολοκλήρωση

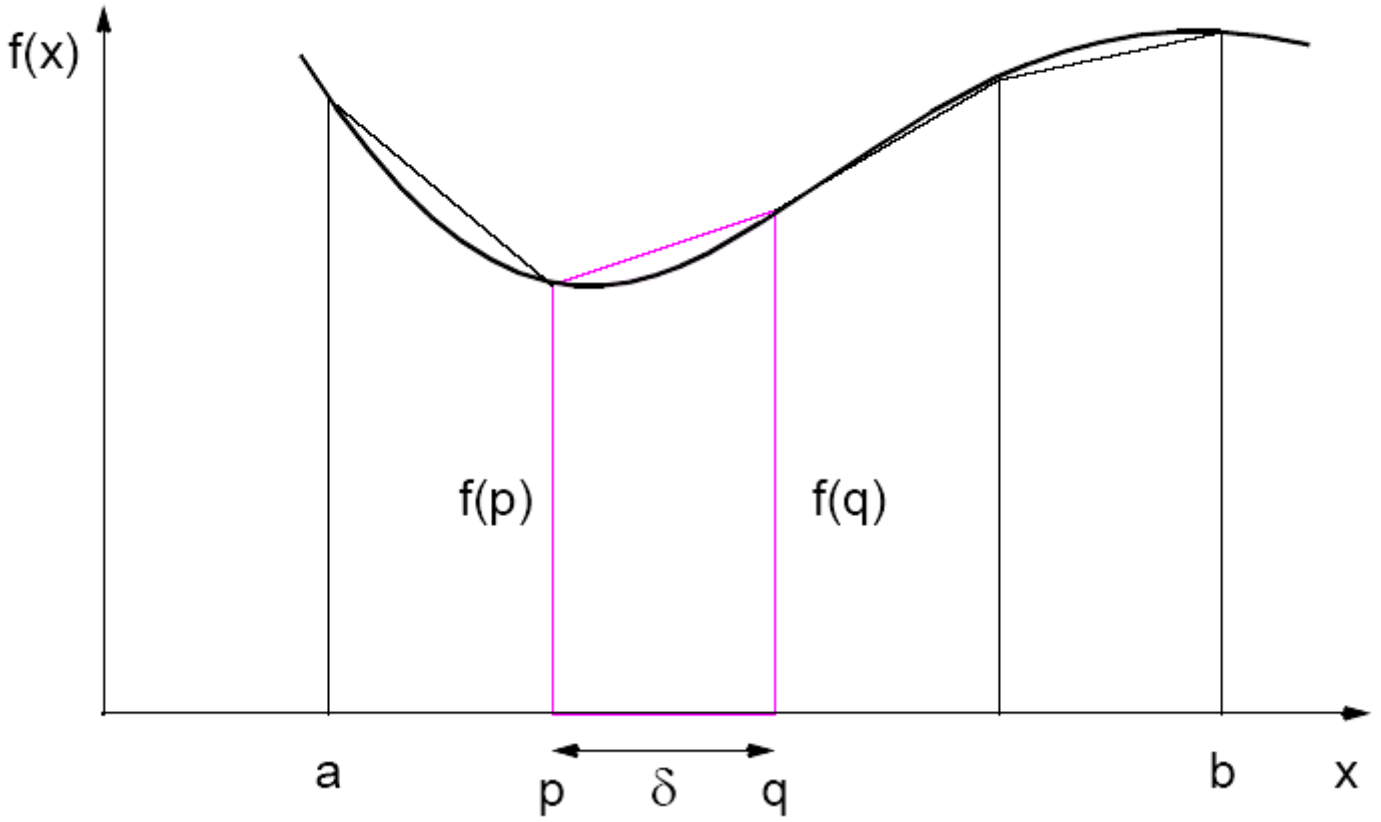
- Υπολογισμός “επιφάνειας κάτω από μια καμπύλη”.
- Επιμερισμός της επιφάνειας σε μικρά τμήματα (παραλληλισμός).
- Μπορεί να εφαρμοστεί και divide and conquer, δηλ. αναδρομικός επιμερισμός της επιφάνειας.

Αριθμητική ολοκλήρωση με ορθογώνια

Το εμβαδό κάθε τμήματος προσεγγλιζεται με το εμβαδό αντίστοιχου ορθογωνίου.



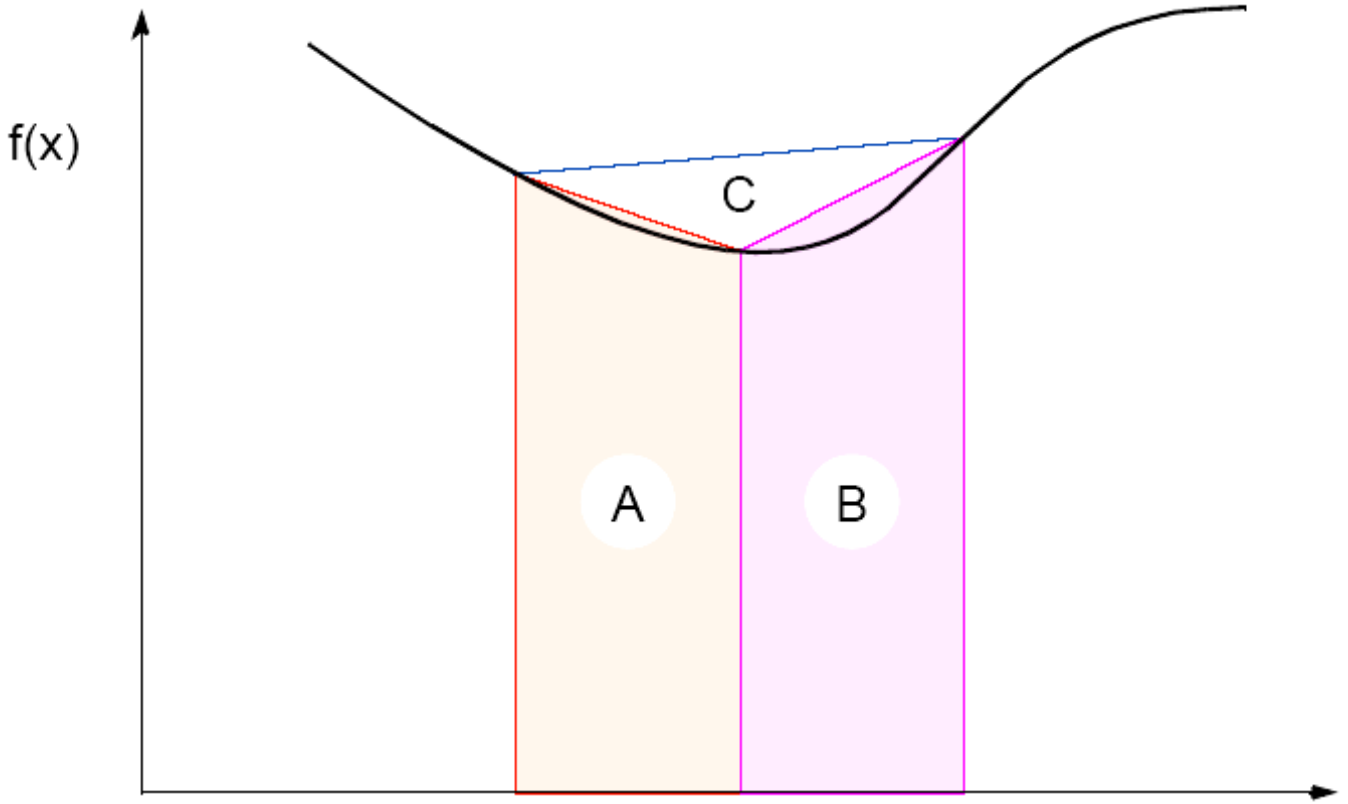
Αριθμητική ολοκλήρωση: κανόνας τραπεζίου



Συγκρίσιμο με τη προηγούμενη μέθοδο, για μικρό βήμα.

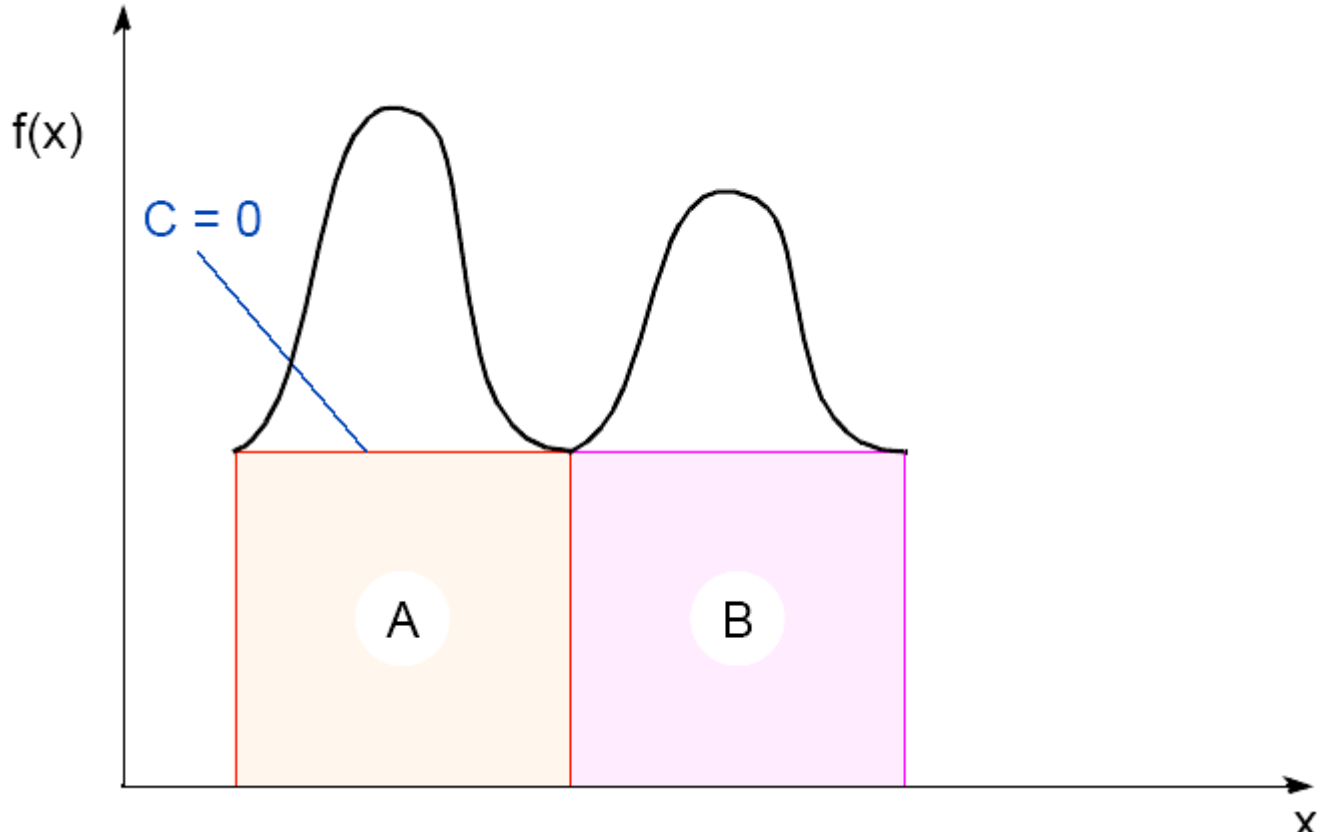
Προσαρμοστικός τετραγωνισμός

Το βήμα προσαρμόζεται στη κλήση της καμπύλης. Γίνεται χρήση τριών επιφανειών, *A*, *B*, και *C*. Τερματισμός για max(A, B) ~ min(A, B) + C ή C ~ 0.



Εσφαλμένος τερματισμός

Προσοχή στο επιλεγόμενο βήμα.



Μπορεί να οδηγήσει σε εσφαλμένο τερματισμό ($C = 0$).

Πρόγραμμα MPI για τον υπολογισμό τού π

```
#include <math.h>                          /*include files*/
#include <stdio.h>
#include "mpi.h

void printit();                          /*function prototypes*/


int main(int argc, char *argv[])
{
double actual_pi = 3.14159265358979323846264643;
                                          /*for comparison*/

int n, rank, num_proc, i;
double temp_pi, calc_pi, int_size, part_sum, x;
char response = 'y', resp1 = 'y';
MPI_Init(&argc, &argv);
```

```
MPI_Comm_size(MPI_COMM_WOLD, &num_proc);
MPI_Comm_rank(MPI_COMM_WOLD, &rank);
if (rank == 0) printit();
                    /*I am master, print out welcome*/

while (response == 'y') {
  if (resp1 == 'y') {
    if (rank == 0) {                    /*I am master*/
     printf ("\nEnter no. of intervals: (0 will
                exit)\n";
     scanf("%d",&n);
    }
  }
  else n = 0;

  MPI_Bcast(&n, 1, MPI_INT, 0);       /*broadcast n*/
  if (n==0) break;       /*check for quit condition*/
```

```
else {
  int_size = 1.0/(double) n;       /*interval size*/
  part_sum = 0.0;

  for (i = rank + 1; i <= n; i += num_proc) {
                                   /* partial sums */
    x = int_size * ((double)i - 0.5);
    part_sum += (4.0 / (1.0 + x*x));
  }
  temp_pi = int_size * part_sum;

          /* collect all partial sums compute pi */
  MPI_Reduce (&temp_pi, &calc_pi, 1, MPI_DOUBLE
    MPI_SUM, 0);
```

```
  if (rank == 0) {                          /*I am master*/
    printf ("\n pi is approximately %f", calc_pi);
    printf ("\n Error is %f",
             fabs(calc_pi - actual_pi);
  }
 }                                           /* end else */
 if (rank == 0) {                          /*I am master*/
   printf ("\nCompute with new intervals?
            (y/n) :");
   scanf ("%c",&resp1);
 }
}                                           /*end while*/
MPI_Finalize();

return 0;
}                                            /*end main*/
```

```
/* functions */

void printit() {
printf ( "\n*********************************" );
printf ( "\nWelcome to the pi calculator!" );
printf ( "\nYou set the number of divisions" );
printf ( "\nfor estimating the integral:" );
printf ( "\n\tf(x)=4/(1+x^2)" );
printf ( "\n*********************************" );
}                                    /*end printit*/
```

Πρόβλημα Ν-Σωμάτων

Εύρεση θέσεων και κινήσεων ουρανίων σωμάτων που υφίστανται βαρυτικές έλξεις από άλλα σώματα, με χρήση των νόμων του Νεύτονα.

Εξισώσεις προβλήματος Ν-σωμάτων

Η βαρυτική έλξη μεταξύ δύο σωμάτων με μάζες m_a and m_b είναι:

$$F = \frac{Gm_a m_b}{r^2}$$

όπου *G* η σταθερά της βαρύτητας και *r* η απόσταση μεταξύ των δύο σωμάτων. Σύμφωνα με το 2ο νόμο του Νεύτονα το κάθε σώμα κινείται με ταχύτητα που ορίζεται από:

$$F = ma$$

όπου *m* είναι η μάζα του σώματος, *F η δύναμη που υφίσταται και α* η επιτάχυνση που προκύπτει.

Λεπτομέρειες

Έστω το χρονικό διάστημα *Δt* . Για σώμα μάζας *m*, η δύναμη είναι:

$$F = \frac{m(v^{t+1} - v^{t})}{\Delta t}$$

Η νέα ταχύτητα:

$$v^{t+1} = v^{t} + \frac{F\Delta t}{m}$$

όπου v^{t+1} είναι η ταχύτητα τη χρονική στιγμή $t + 1$ και v^{t} είναι η ταχύτητα τη χρονική στιγμή *t*. Στο χρονικό διάστημα *Δt η θέση αλλάζει:*

$$x^{t+1} - x^{t} = v\Delta t$$

όπου x^{t} είναις η θέση του σώματος τη χρονική στιγμή *t*.
Μετά τη μετακίνηση των σωμάτων οι δυνάμεις αλλάζουν, άρα οι υπολογισμοί πρέπει να επαναληφθούν.

Ακολουθιακός κώδικας

Overall gravitational *N*-body computation can be described by:

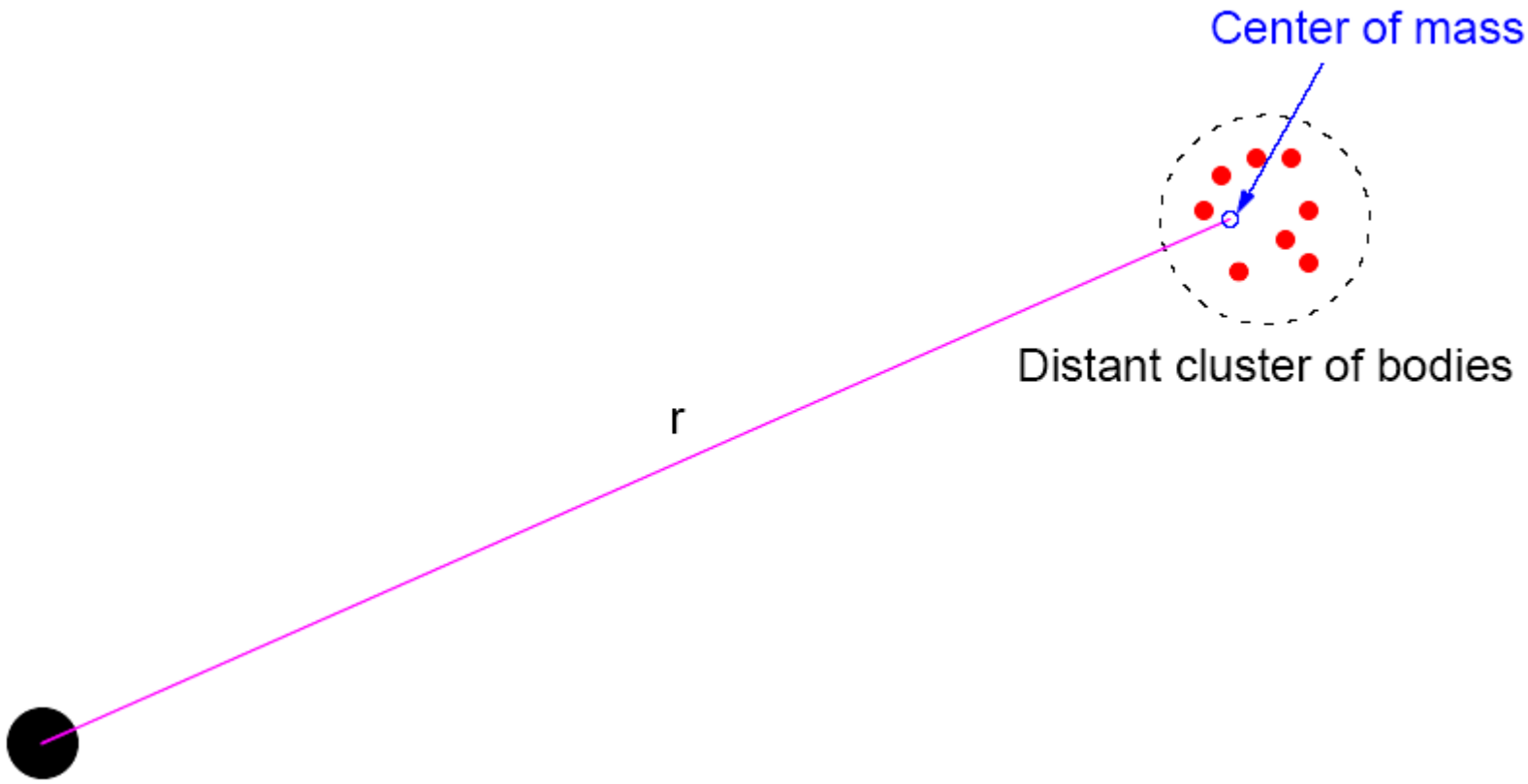
```
/* for each time period */
for (t = 0; t < tmax; t++){

  for (i = 0; i < N; i++) {          /*for each body*/
    F = Force_routine(i);            /*force on body*/
    v[i]new = v[i] + F * dt / m;        /*velocity*/
    x[i]new = x[i] + v[i]new * dt;      /*position*/

  }
  for (i = 0; i < N; i++) {       /* for each body */
      x[i] = x[i]new;             /*update position*/
      v[i] = v[i]new;             /*update velocity*/
  }
}
```

Παράλληλος κώδικας

Ο ακολουθιακός αλγόριθμος απαιτεί $O(N^2)$ πράξεις (για κάθε επανάληψη) καθώς κάθε ένα από τα N σώματα επηρρεάζεται από τα άλλα N - 1 σώματα.

Ο αλγόριθμος είναι αποδοτικός μόνο για σχετικά μικρά N αλλά στα ενδιαφέροντα προβλήματα το N είναι πολύ μεγάλο.

Ο φόρτος μπορεί να μειωθεί με την αντικατάσταση μιας συστοιχίας απομακρυσμένων σωμάτων από ένα μόνο σώμα με μάζα ίση με το άθροισμα των μαζών και θέση στο κέντρο μάζας της συστοιχίας:



Αλγόριθμος Barnes-Hut

Αρχικά ένας κύβος περιλαμβάνει όλο το χώρο που έχει σώματα.

- Πρώτα ο κύβος διαιρείται σε οκτώ υπο-κύβους.
- Αν ένας υπο-κύβος δεν περιέχει σώματα διαγράφεται.
- Αν ένας υπο-κύβος έχει μόνο ένα σώμα παραμένει.
- Αν ένας υπο-κύβος έχει δύο ή περισσότερα σώματα διαιρείται αναδρομικά, μέχρι να μείνουν υπο-κύβοι που έχουν μόνο ένα σώμα.

Αλγόριθμος Barnes-Hut (συν.)

Δημιουργεί *οκταδικό δένδρο (octtree)* – κάθε κόμβος έχει μέχρι οκτώ κλάδους.

Τα φύλλα είναι υπο-κύβοι του ενός σώματος.

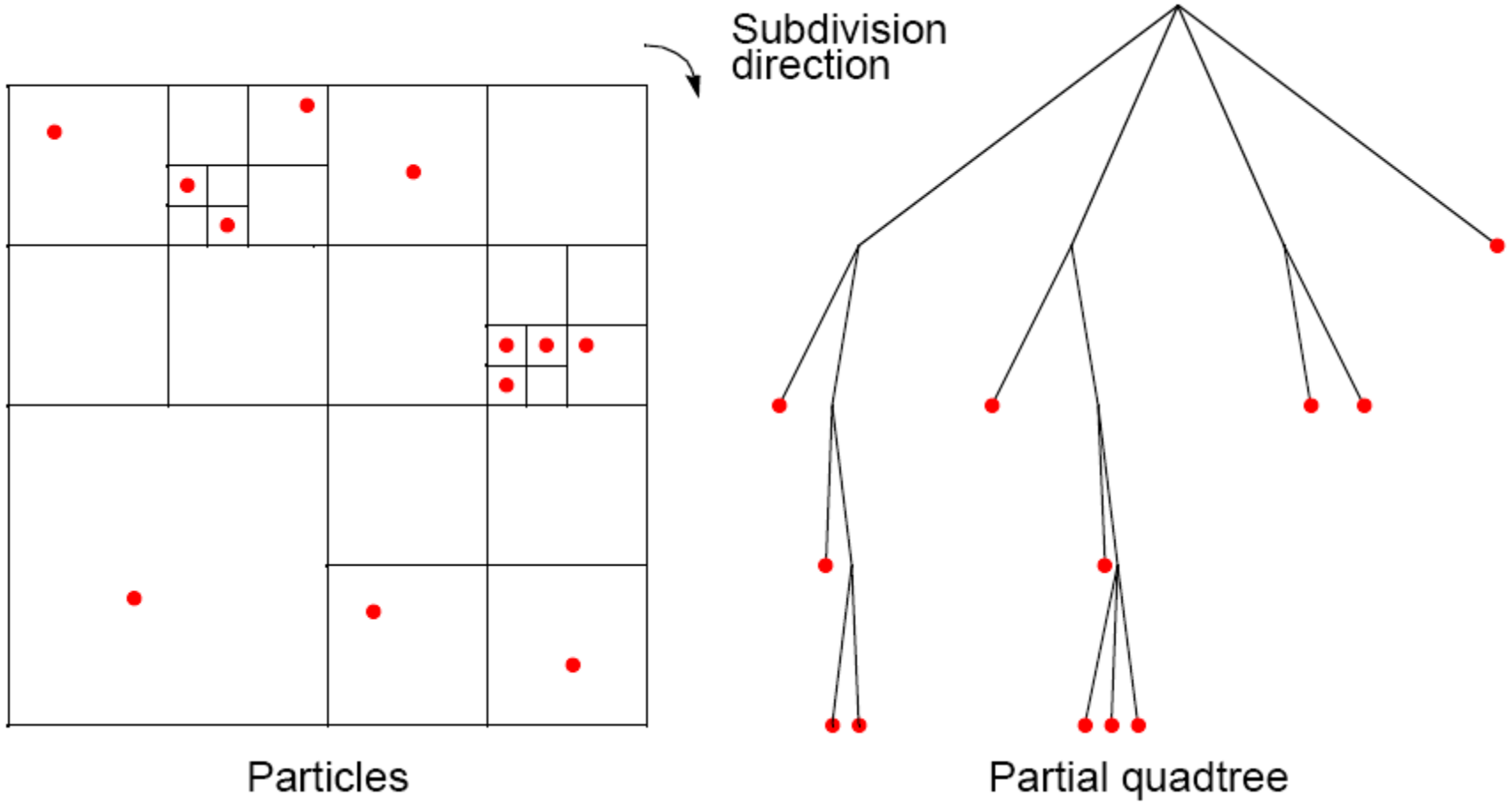
Η συνολική μάζα και το κέντρο μάζας κάθε υπο-κύβου αποθηκεύεται σε κάθε κόμβο.

Ξεκινάμε από τα φύλλα και προχωρούμε προς τη ρίζα..

Η δύναμη που δέχεται κάθε σώμα υπολογίζεται με διάσχιση του δένδρου, ξεκινώντας από τη ρίζα και σταματώντας στους κόμβους με την επιθυμητή ανάλυση, πχ αν η απόσταση r του σώματος από το κέντρο μάζας ενός υπο-κύβου είναι μεγαλύτερη από την ακμή του υπο-κύβου.

Η κατασκευή του δένδρου απαιτεί χρόνο O($n$log$n$), και ο υπολογισμός όλων των δυνάμεων άλλο τόσο, άρα η συνολική πολυπλοκότητα της μεθόδου είναι O($n$log$n$).

Αναδρομική διαίρεση στις 2 διαστάσεις.



Ορθογωνική αναδρομική διχοτόμηση

Για διδιάστατο χώρο: Βρίσκουμε μια κατακόρυφη γραμμή που χωρίζει το χώρο σε δύο περιοχές με ίσο αριθμό σωμάτων. Σε κάθε περιοχή βρίσκουμε μια οριζόντια γραμμή που χωρίζει τη περιοχή σε δύο υπο-περιοχές με ίσο αριθμό σωμάτων. Συνεχίζουμε με εναλλαγή κατακόρυφων και οριζόντιων γραμμών, όσο απαιτείται.