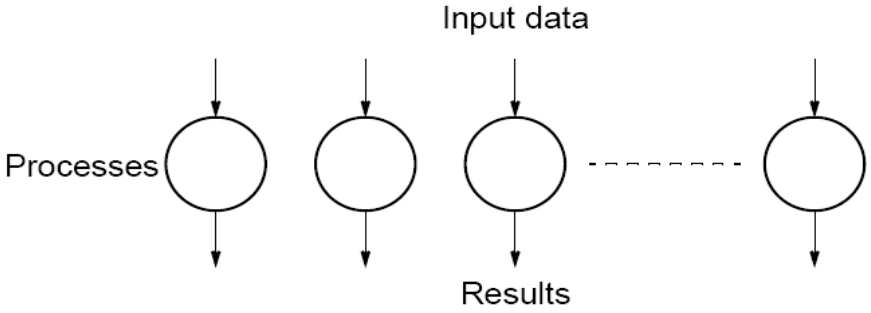
Embarrassingly Parallel Computations (Πολύ εύκολες παραλληλοποιήσεις)

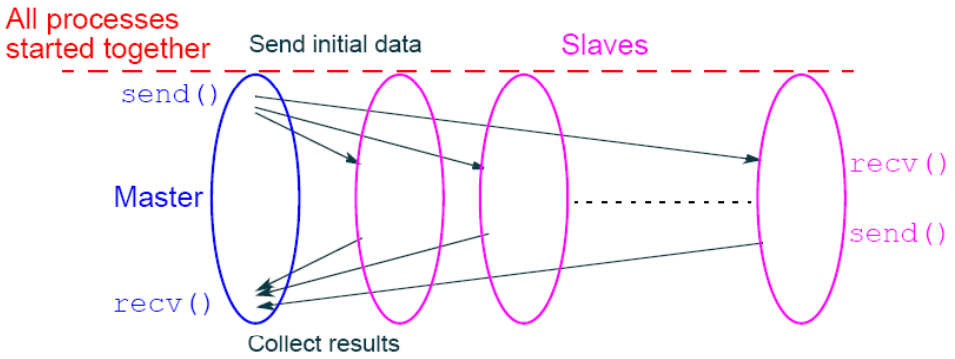
Πολύ εύκολες παραλληλοποιήσεις

Υπάρχουν παραδείγματα υπολογισμών οι οποίοι μπορούν με προφανή τρόπο να διαιρεθούν σε τελείως ανεξάρτητα έργα. Καθένα από αυτά τα έργα μπορούν να εκτελεστούν από ξεχωριστές διεργασίες (επεξεργαστές)



Οι διεργασίες είτε δεν επικοινωνούν καθόλου ή η επικοινωνία τους είναι ελάχιστη

Κάθε διεργασία εκτελεί τα έργα της χωρίς αλληλεπίδραση με τις άλλες διεργασίες



Usual MPI approach

Συνήθως αυτού του είδους οι υπολογισμοί, υλοποιούνται με την τεχνική master – slave.
Αρχικά, η διεργασία master μοιράζει τα δεδομένα εισόδου σε ένα πλήθος slave διεργασιών
Κάθε διεργασία slave υλοποιεί ένα τμήμα του υπολογισμού χωρίς να επικοινωνεί με άλλες διεργασίες
Με την ολοκλήρωση του υπολογισμού που τις έχει ανατεθεί, κάθε διεργασία slave στέλνει το τοπικό αποτέλεσμα στη διεργασία master
Η διεργασία master συνδυάζει όλα τα αποτελέσματα από τις slave διεργασίες και υπολογίζει το τελικό αποτέλεσμα

Πολύ Εύκολες Παραλληλοποιήσεις

- Επεξεργασία εικόνας χαμηλού επιπέδου
- Σύνολο Mandelbrot
- Υπολογισμοί Monte Carlo

Επεξεργασία εικόνας χαμηλού επιπέδου

- Πολλές λειτουργίες από την περιοχή της επεξεργασίας εικόνας είναι τοπικές, δηλ. το αποτέλεσμα σε κάθε pixel της εικόνας προκύπτει ως συνάρτηση της τρέχουσας τιμής του pixel ή/και των γειτονικών pixels.
- Αν η εικόνα διαιρεθεί σε περιοχές και κάθε περιοχή ανατεθεί σε ξεχωριστή διεργασία, οι διεργασίες μπορούν να επεξεργαστούν τις τοπικές τους περιοχές χωρίς να επικοινωνούν μεταξύ τους.

Μερικοί γεωμετρικοί μετασχηματισμοί

Ακολουθούν κάποιες συχνές λειτουργίες επεξεργασίας εικόνας σε χαμηλό επίπεδο

Ολίσθηση:
Το αντικείμενο ολισθαίνει κατά D*x* κατά μήκος του άξονα x και κατά D*y* κατά μήκος του άξονα *y*:

$$x¢ = x + Dx$$
$$y¢ = y + Dy$$

όπου *x* και *y* είναι οι αρχικές συντεταγμένες και *x¢*, *y¢* είναι οι νέες συντεταγμένες.

Κλιμάκωση:
Τα αντικείμενο κλιμακώνεται με συντελεστή *Sx* στον άξονα των x και με συντελεστή *Sy* στον άξονα των *y:*

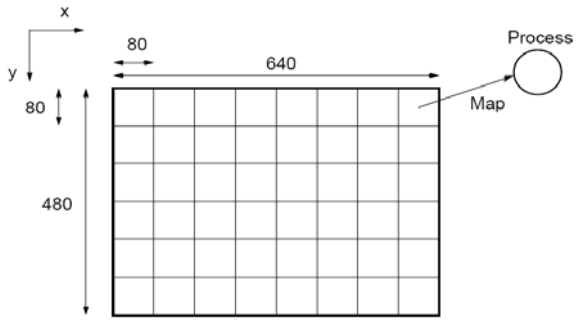
$$x¢ = xSx$$
$$y¢ = ySy$$

Περιστροφή:
Το αντικείμενο περιστρέφεται κατά μία γωνία q γύρω από την αρχή των αξόνων:

$$x¢ = x \cos q + y \sin q$$
$$y¢ = -x \sin q + y \cos q$$

Όλοι αυτοί οι υπολογισμοί μπορούν εύκολα να παραλληλοποιηθούν χωρίς οι παράλληλές διεργασίες να επικοινωνούν μεταξύ τους.

Διαμοιρασμός μίας εικόνας στις διεργασίες



Η εικόνα διαιρείται σε ορθογώνιες περιοχές (80x80) και κάθε διεργασία επεξεργάζεται την περιοχή που τις αντιστοιχεί
Εναλλακτικά, η εικόνα μπορεί να διαιρεθεί σε οριζόντιες περιοχές (strips)

Mandelbrot Set

Το σύνολο Mandelbrot είναι ένα σύνολο σημείων στο μιγαδικό χώρο τα οποία είναι ημισταθερά δηλ. αυξάνονται και μειώνονται χωρίς να υπερβαίνουν ένα συγκεκριμένο όριο και υπολογίζονται επαναληπτικά με βάση τη συνάρτηση

$$z_{k+1} = z_k^2 + c$$

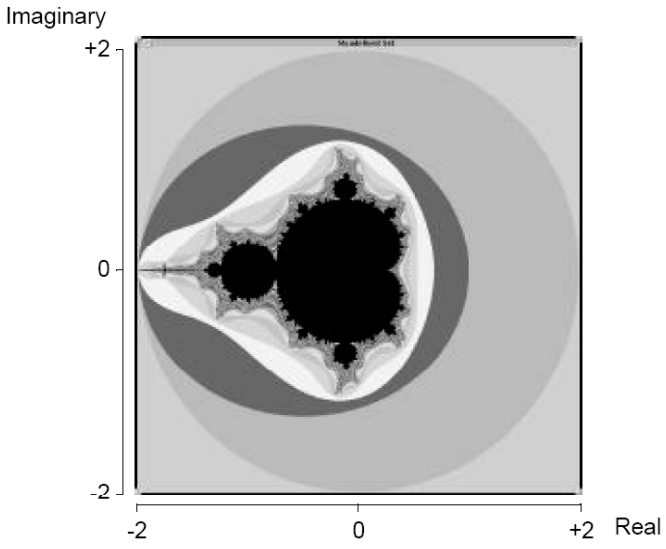
όπου z_{k+1} είναι ο μιγαδικός αριθμός $z = a + bi$ που προκύπτει μετά από την $(k + 1)$-οστη επανάληψη και c είναι ένας μιγαδικός αριθμός ο οποίος δίνει τη θέση του σημείου στο επίπεδο των μιγαδικών. Η αρχική τιμή του z είναι μηδέν.
Οι επαναλήψεις συνεχίζονται μέχρι το μέγεθος του z υπερβεί το 2 ή το πλήθος των επαναλήψεων φτάσει ένα συγκεκριμένο όριο. Το μέγεθος του z είναι το μήκος του διανύσματος και δίνεται από την ακόλουθη σχέση:

$$z_{length} = \sqrt{a^2 + b^2}$$

Ακολουθεί η ακολουθιακή διαδικασία για τον υπολογισμό της τιμής σε ένα σημείο. Επιστρέφει το πλήθος των επαναλήψεων.

```
structure complex {
float real;
float imag;
};
int cal_pixel(complex c)
{
int count, max;
complex z;
float temp, lengthsq;
max = 256;
z.real = 0; z.imag = 0;
count = 0;                          /* number of iterations */
do {
temp = z.real * z.real - z.imag * z.imag + c.real;
z.imag = 2 * z.real * z.imag + c.imag;
z.real = temp;
lengthsq = z.real * z.real + z.imag * z.imag;
count++;
} while ((lengthsq < 4.0) && (count < max));
return count;
}
```

Το σύνολο Mandelbrot



• Το αποτέλεσμα της εκτέλεσης του προηγούμενου κώδικα φαίνεται στο σχήμα. Οι συντεταγμένες κάθε pixel είναι το πραγματικό και φανταστικό τμήμα του μιγαδικού c που περνάει ως όρισμα στη cal_pixel

• Το πλήθος των επαναλήψεων που επιστρέφονται καθορίζουν και το χρώμα του αντίστοιχου pixel.

Παράλληλη Υλοποίηση του υπολογισμού του συνόλου Mandelbrot

Στατική καταχώρηση Έργων

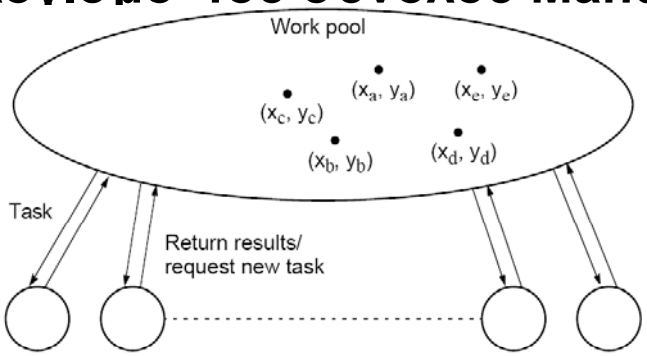
• Σε αυτή τη προσέγγιση, η περιοχή διαιρείται σε ένα σταθερό πλήθος περιοχών και κάθε διεργασία αναλαμβάνει τον υπολογισμό σε μία τέτοια περιοχή.
• Δεν είναι καλύτερη προσέγγιση για το συγκεκριμένο πρόβλημα αφού διαφορετικές περιοχές απαιτούν διαφορετικό πλήθος επαναλήψεων και επομένως χρόνο.
• Αυτό έχει ως αποτέλεσμα κάποιοι επεξεργαστές να εργάζονται ενώ άλλοι να είναι αδρανείς έχοντας τελειώσει νωρίτερα την εργασία τους

Δυναμική καταχώρηση Έργων

.

• Χρειάζεται να επιτευχθεί εξισορρόπηση φορτίου σε όλες τις διεργασίες.
• Η δυναμική καταχώρηση έργων διατηρεί μία «δεξαμενή» με τα προς εκτέλεση έργα. Κάθε διεργασία αναλαμβάνει την εκτέλεση του επόμενου διαθέσιμου έργου. Όταν, μία διεργασία ολοκληρώσει το έργο που έχει αναλάβει, ζητά το επόμενο έργο από τη δεξαμενή και η διαδικασία αυτή επαναλαμβάνεται μέχρι η δεξαμενή να «αδειάσει» από έργα.
• Συνήθως, η master διεργασία αναλαμβάνει τη διαχείριση της δεξαμενής έργων

Δυναμική καταχώριση έργων για τον Υπολονισμό του συνόλου Mandelbrot



• Στο συγκεκριμένο υπολογισμό, κάθε έργο αντιστοιχεί στο υπολογισμό που γίνεται σε ένα pixel της εικόνας.
• Λόγω του μικρού όγκου υπολογισμού (fine-grained υπολογισμός) που αναλαμβάνει κάθε φορά μία διεργασία, ζητάει συχνά επιπλέον έργα για να εκτελέσει με αποτέλεσμα τη συχνή επικοινωνία μεταξύ master και slave διαδικασιών.
• Για αυτό το λόγο, τα έργα που μοιράζονται στις slave διεργασίες περιλαμβάνουν συνήθως περισσότερους υπολογισμούς. Συγκεκριμένα, κάθε έργο είναι οι υπολογισμοί που πρέπει να γίνουν για τα pixels μίας γραμμής της εικόνας.

Ψευδοκώδικας για τον παράλληλο υπολογισμό του συνόλου Mandelbrot

```
Master
count=0;
row=0;
for (k=0; k< num_proc; k++)
   send(&row, P_k, data_tag);
   count++
   row++
}
do {
   recv(&slave, &r, color, P_ANY, result_tag);
   count--;
   if (row < disp_height) {
       send( &row, P_slave, data_tag);
         row++;
         count++;
   } else
     send( &row, P_slave,terminator_tag);
   display(r, color);
} while (count >0)
```

```
Slave
recv(y, P_master, source_tag);
while (source_tag == data_tag) {
c.imag = imag_min + ((float) y *
scale_imag);
for (x=0; x < disp_width; x++) {
   c.real = real_min + ((float) x
*scale_real);
   color[x]=cal_pixel(c);
}
send(&x, &y, color, Pmaster,
result_tag);
recv(&y, P_master, source_tag);
}
```

Μέθοδοι Monte Carlo

• Οι μέθοδοι Monte Carlo χρησιμοποιούν τυχαίες επιλογές στους υπολογισμούς. Βρίσκουν εφαρμογή στη λύση αριθμητικών υπολογισμών και προβλημάτων φυσικής.

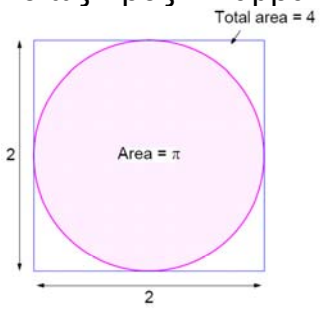
• Οι υπολογισμοί που αντιστοιχούν στις τυχαίες επιλογές μπορούν να γίνουν ανεξάρτητα ο ένας από τον άλλο και επομένως και οι μέθοδοι Monte Carlo είναι υπολογισμοί με πολύ εύκολη παραλληλοποίηση.

Παράδειγμα – υπολογισμός του π

• Ένα παράδειγμα Monte Carlo μεθόδου είναι ο υπολογισμός του π.
• Συγκεκριμένα, αν υποθέσουμε ένα κύκλο ακτίνας ίσης με 1 εντός ενός τετραγώνου 2x2, τότε ο λόγος της εμβαδού του κύκλου προς το εμβαδό του τετραγώνου είναι:

$$\frac{\text{Area of circle}}{\text{Area of square}} = \frac{\pi(1)^2}{2 \times 2} = \frac{\pi}{4}$$

• Εκτελούμε ένα τυχαίο πείραμα επιλέγοντας σημεία μέσα στο τετράγωνο με τυχαίο τρόπο. Στη συνέχεια μετρούμε το πλήθος των περιπτώσεων που το τυχαία επιλεγμένο σημείο είναι εντός του κύκλου.
• Το ποσοστό f των σημείων εντός κύκλου θα συγκλίνει στο π/4 μετά από αρκετές επαναλήψεις του πειράματος δηλ f = π/4
• Επιλύοντας τον παραπάνω τύπο ως προς π λαμβάνουμε μία προσεγγιστική τιμή για το π



Υπολογισμός Ολοκληρώματος με τη μέθοδο Monte Carlo

Για τον υπολογισμό του ολοκληρώματος $\int_a^b f(x)dx$ υπολογίζουμε τη τιμή της συνάρτησης f σε τυχαία σημεία $x_1, x_2, .., x_N$ στο διάστημα [a ... b]

Στη συνέχεια, το ολοκλήρωμα θα είναι ίσο με $\lim_{N\to\infty} \frac{1}{N} \sum_{i=1}^{N} f(x_i)(b-a)$

Η μέθοδος Monte Carlo είναι πολύ χρήσιμη εάν δεν μπορεί να βρεθεί το ολοκλήρωμα της συνάρτησης με κάποια αριθμητική μέθοδο όταν π.χ. περιέχει ένα μεγάλο αριθμό μεταβλητών

Για παράδειγμα για τον υπολογισμό του ολοκληρώματος $\int_{x_1}^{x_2} \left(x^2 - 3x\right)dx$ ο ακολουθιακός κώδικας θα είναι:

```
sum = 0;
for (i = 0; i < N; i++) {          /* N random samples */
xr = rand_v(x1, x2);               /* generate next random value */
sum = sum + xr * xr - 3 * xr;      /* compute f(xr) */
}
area = (sum / N) * (x2 - x1);
```

Στη παράλληλη υλοποίηση της μεθόδου Monte Carlo, κάθε slave διεργασία αναλαμβάνει τον υπολογισμό της συνάρτησης σε ένα ή περισσότερα σημεία και στη συνέχεια αθροίζουν τις τιμές της συνάρτησης. Η master διεργασία αναλαμβάνει να αθροίσει τα μερικά αθροίσματα που τις στέλνουν οι slave διεργασίες σύμφωνα με το παραπάνω τύπο

Ψευδοκώδικας για την παράλληλη υλοποίηση της ολοκλήρωσης Monte Carlo

```
Master
for (i=0; i<N/n; i++){
   for (j=0; j<n; j++)
        xr[j]=rand();
   recv(P_ANY,req_tag,P_source);
   send(xr, &n, P_source,compute_tag);
}
for (i=0; i<num_slaves;i++){
   recv(P_i, req_tag);
   send(P_i, compute-tag);
}
sum = 0;
reduce_add(&sum, P_group);
```

```
Slave
sum=0;
send(P_master, req_tag);
recv(xr, &n, P_master, source_tag);
while (source_tag == compute_tag){
   for (i=0; i < n; i++)
        sum = sum +xr[i]*xr[i] -
        3*xr[i];
   send(P_master, req_tag);
   recv(xr, &n, P_master, source_tag);
}
reduce_add(&sum, P_group);
```