ΠΑΝΕΠΙΣΤΗΜΙΟ
ΔΥΤΙΚΗΣ ΜΑΚΕΔΟΝΙΑΣ

Συστήματα Παράλληλης και Κατανεμημένης Επεξεργασίας

Ενότητα: ΕΡΓΑΣΤΗΡΙΑΚΗ ΑΣΚΗΣΗ No:04

Δρ. Μηνάς Δασυγένης

mdasyg@ieee.org

Τμήμα Μηχανικών Πληροφορικής και Τηλεπικοινωνιών

Εργαστήριο Ψηφιακών Συστημάτων και Αρχιτεκτονικής Υπολογιστών

http://arch.icte.uowm.gr/mdasyg

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 του Πανεπιστημίου 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.

Περιεχόμενα

1. Σκοπός της άσκησης

- Συγχρονισμός ώρας των κόμβων μέσω NTP.
- Είσοδος από το πληκτρολόγιο και απλή αριθμητική επεξεργασία.
- Τεχνικές αποφυγής αδιεξόδου MPI_Send, MPI_Recv.
- Άσκηση κατανεμημένων υπολογισμών.

2. Παραδοτέα

(A) 2 ερωτήσεις

(C) 5 ασκήσεις

Από το 2017 η ενότητα 3 είναι προαιρετική και το εργαστήριο μπορεί να ξεκινήσει από την ενότητα 4, εφόσον χρησιμοποιείται το MTL και το σύστημα υποβολής εργασιών PBS. Η ενότητα 3 απαιτεί προσωπικό VPS και προϋποθέτει ότι έχουν υλοποιηθεί οι προαιρετικές ρυθμίσεις VPS των προηγούμενων εργαστηρίων.

3. Συγχρονισμός ώρας των κόμβων

Είναι πολύ σημαντικό όλοι οι κόμβοι της συστοιχίας να έχουν την ίδια ώρα και να μην παρουσιάζονται αποκλίσεις χρόνου. Αυτό θα βοηθήσει, ώστε να μπορεί να δημιουργηθεί μια σωστή σειρά των μηνυμάτων. Διαφορετικά ο απο-συγχρονισμός ώρας σε μερικές περιπτώσεις μπορεί να οδηγήσει σε πολύ δυσάρεστες καταστάσεις. Για να επιτευχθεί ο συγχρονισμός ώρας θα χρησιμοποιήσουμε το πρόγραμμα NTP *(network time protocol).* Το NTP ρωτάει έναν κοντινό κεντρικό διακομιστή την ώρα και στη συνέχεια λαμβάνοντας υπόψιν την καθυστέρηση μετάδοσης λόγο του δικτύου ρυθμίζει τον υπολογιστή μας στην ώρα που έχει ο διακομιστής. Ο κοντινότερος διακομιστής στους κόμβους μας είναι ο ntp.uowm.gr.

Για να ελεγχθεί η διαφορά ρολογιού θα εκτελεστεί η εντολή:

```
/usr/sbin/ntpdate  -q ntp.uowm.gr
```

Όταν εκτελείτε την εντολή θα σας εμφανιστεί η απόκλιση. Σημειώστε το stratum του ntp.uowm.gr, το offset, και το delay. _____

Επιβεβαιώστε αν εκτελείται το πρόγραμμα ntp με την εντολή

```
ps axuw | grep ntp
```

Αν δείτε ότι εκτελείται τότε η ώρα στο μηχάνημά σας διατηρείτε ενημερωμένη και μπορείτε να προσπεράσετε αυτή την παράγραφο.

Αν δεν εκτελείτε η εντολή ntp, τότε δώστε

```
/usr/sbin/ntpdate -s ntp.uowm.gr
```

Εκτελέστε την ανωτέρω εντολή και επιβεβαιώστε ότι έγινε ο συγχρονισμός ώρας με το να δείτε το αρχείο καταγραφής μηνυμάτων /var/log/messages ως εξής για το ΛΣ FreeBSD

```
tail /var/log/messages
```

ή για το Ubuntu Linux

```
tail /var/log/syslog
```

Επιβεβαιώστε ότι έγινε η ρύθμιση της ώρας με το να υπάρχει η λέξη `...adjust...`

Μάλιστα προκειμένου να διατηρείται ο συγχρονισμός θα ρυθμίσουμε τον κόμβο να εκτελεί το πρόγραμμα συγχρονισμού κάθε 1 ώρα. Αυτό θα γίνει με το να ρυθμίσουμε το πρόγραμμα CRON.

1. Διαβάστε το εγχειρίδιο χρήσης του προγράμματος CRON *(κυρίως τις παραγράφους NAME και DESCRIPTION)* με την εντολή:

   `man cron`

2. Δημιουργήστε ένα αρχείο /tmp/croninfo.txt
3. Τοποθετήστε τη γραμμή:

   `@hourly /usr/sbin/ntpdate -s ntp.uowm.gr`

4. Αποθηκεύεστε το αρχείο.
5. Εισάγετε το αρχείο στο σύστημα CRON ως εξής:

   `crontab /tmp/croninfo.txt`

6. Επιβεβαιώστε ότι έχετε εισάγει στο σύστημα CRON τις ρυθμίσεις με το να εμφανίσετε τις τρέχουσες προγραμματισμένες εργασίες ως εξής:

   `crontab -l`

7. Από εδώ και πέρα κάθε ώρα ο κόμβος σας θα συγχρονίζεται με την κεντρική ώρα του Πανεπιστημίου που βρίσκεται στο **`ntp.uowm.gr`**

4. Είσοδος από το πληκτρολόγιο και απλή αριθμητική επεξεργασία

5. Αντιγράψτε το αρχείο **send_receive3.c** στο **send_receive4.c**.
6. Τροποποιήστε το πρόγραμμα ώστε το rank0 να ζητάει *(με τη χρήση της συνάρτησης* `scanf()`*)* από το χρήστη να πληκτρολογήσει μια ακέραια τιμή από 0 έως 99.

7. Σε περίπτωση που ο χρήστης πληκτρολογήσει κάτι που δεν ανήκει στο παραπάνω εύρος θα πηγαίνει το rank0 στο προηγούμενο βήμα *(της προτροπής και της επανα-εισαγωγής αριθμού).*
8. Στη συνέχεια το rank0 θα στέλνει αυτήν την ακέραια τιμή στο επόμενο rank *(αποστολή στο* **`rank+1`***).*
9. Το επόμενο rank θα αυξάνει την τιμή που έδωσε ο χρήστης κατά βήμα όσο είναι το rank, θα εκτυπώνει την τιμή που έλαβε, την τιμή που έστειλε και θα τη στέλνει στο επόμενο rank *(αποστολή στο* **`rank+1`***).*
10. Η διεργασία με το τελευταίο rank (rank==communicator size) θα αυξάνει την τιμή που έλαβε με όσο rank έχει, και θα στέλνει την τιμή στο rank0.
11. Το rank0 θα λαμβάνει την τιμή αυτή και θα εκτυπώνει ένα μήνυμα τέλους μαζί με την τιμή που έχει αυτή τη στιγμή.
12. Θεωρήστε ότι απαιτούνται τουλάχιστον 2 διεργασίες, οπότε σε περίπτωση που το communicator size είναι μικρότερο από 2, να εμφανίζεται ένα μήνυμα **`"Minimum 2 processes are required"`** και να καλείται η MPI_Abort.
13. **(C1)** Να δοθεί το πηγαίο αρχείο.
14. Μεταγλωττίστε το πρόγραμμα και εκτελέστε το για 1, 2, 3, 10 διεργασίες και επαληθεύστε τη σωστή λειτουργία.
15. **(C2)** Να δώσετε το κατάλληλο sreenshot εκτέλεσης για 10 διεργασίες.

5. Αποφυγή Αδιεξόδων

16. Δημιουργήστε ένα αρχείο **send_receive_dlock.c** στο οποίο τοποθετήστε τον παρακάτω κώδικα:

```
#include <stdio.h>
#include <mpi.h>
int main (int argc, char* argv[])
{
int rank, count;
MPI_Status status;
double msg1[50], msg2[50];
MPI_Init(&argc, &argv);
MPI_Comm_rank(MPI_COMM_WORLD, &rank);
MPI_Comm_rank(MPI_COMM_WORLD, &rank);
if (rank==0) {
MPI_Recv(msg2, 10, MPI_DOUBLE, 1, 0, MPI_COMM_WORLD,
&status);
MPI_Send (msg1, 50, MPI_DOUBLE, 1, 0, MPI_COMM_WORLD);
}
else if (rank==1) {
```

```
MPI_Recv(msg2, 50, MPI_DOUBLE, 0, 0, MPI_COMM_WORLD,
&status);
MPI_Send (msg1, 10, MPI_DOUBLE, 0, 0, MPI_COMM_WORLD); }
MPI_Finalize();
return (0); }
```

17. **(A1)** Μεταγλωττίστε το πρόγραμμα και εκτελέστε το σε 2 ή περισσότερους κόμβους. Τι παρατηρείτε; Τερματίζει το πρόγραμμα ή όχι;
18. **(A2)** Υπάρχει κάποιο πρόβλημα;
19. Αντιγράψτε το αρχείο **send_receive_dlock.c** στο αρχείο **(C3) send_receive_dlock2.c** και **(C4) send_receive_dlock3.c** και τροποποιήστε το κάθε αρχείο, ώστε να μην εμφανίζεται η δυσάρεστη κατάσταση του προηγούμενου βήματος, ως εξής:
    1. Στο **send_receive_dlock2.c** να αναδιατάξετε τις κλήσεις των συναρτήσεων
    2. στο **send_receive_dlock3.c** να αντικαταστήσετε την κλήση συναρτήσεων με άλλες παρόμοιες *(ίδιας λειτουργικότητας).*
20. Να παραδοθούν τα 2 αρχεία ως **c3.c** και **c4.c**

Βασικοί Κατανεμημένοι Υπολογισμοί

1. Δημιουργήστε ένα νέο αρχείο με όνομα **calc-00.c**
2. Τοποθετήστε τον κατάλληλο κώδικα χρήσης του OpenMPI
3. Θεωρήστε ότι στο αρχείο αυτό υπάρχουν δηλωμένες οι μεταβλητές

```
float a[4] = { 1.0, 2.3, 4.5, 5.9 } ;
float b[4];
```

4. Κατασκευάστε τον κώδικα που να υλοποιεί τα παρακάτω:
    1. Η πρώτη διεργασία θα στέλνει τον πίνακα `a[]` στις υπόλοιπες διεργασίες
    2. Η δεύτερη διεργασία θα λαμβάνει το `a[]`, θα τον τοποθετεί στον πίνακα b, και θα υπολογίζει το άθροισμα των στοιχείων του. Θα χρησιμοποιηθεί ένας βρόχος επανάληψης από `b[0]` έως `b[size]`, όπου **size** θα βρεθεί μέσω της σχετικής συνάρτησης καταμέτρησης δεδομένων, που είδαμε σε προηγούμενη άσκηση.
    3. Η τρίτη διεργασία θα λαμβάνει το `a[]`, θα τον τοποθετεί στον πίνακα b[], και θα υπολογίζει το μέγιστο των στοιχείων του. Θα χρησιμοποιηθεί ένας βρόχος επανάληψης από `b[0]` έως `b[size]`, όπου **size** θα βρεθεί μέσω της σχετικής συνάρτησης καταμέτρησης δεδομένων, που είδαμε σε προηγούμενο εργαστήριο.
    4. Η τέταρτη διεργασία θα λαμβάνει το `a[]`, θα τον τοποθετεί στον πίνακα `b[]`, και θα υπολογίζει το ελάχιστο των στοιχείων του. Θα χρησιμοποιηθεί ένας βρόχος επανάληψης από `b[0]` έως `b[size]`, όπου **size** θα βρεθεί μέσω της σχετικής συνάρτησης καταμέτρησης δεδομένων, που είδαμε σε προηγούμενο εργαστήριο.

5. Η πέμπτη διεργασία θα λαμβάνει το `a[]`, θα τον τοποθετεί στον πίνακα b[], και θα υπολογίζει το γινόμενο των στοιχείων του. Θα χρησιμοποιηθεί ένας βρόχος επανάληψης από `b[0]` έως `b[size]`, όπου **size** θα βρεθεί μέσω της σχετικής συνάρτησης καταμέτρησης δεδομένων, που είδαμε σε προηγούμενο εργαστήριο.

5. Όλες οι διεργασίες θα στέλνουν τον πραγματικό αριθμό που θα έχουν υπολογίσει στην πρώτη διεργασία η οποία θα εκτυπώνει τα μηνύματα σχετικά με το άθροισμα, μέγιστο, ελάχιστο, γινόμενο και τις αντίστοιχες τιμές που έλαβε από τις υπόλοιπες.
6. **(C5)** Να δοθεί ο πηγαίος κώδικας, ως **c5.c**

ΑΣΚΗΣΗ (C6)

Αντιγράψτε το πρόγραμμα **calc-00.c** στο **calc-01.c .**

Τροποποιήστε το **calc-01.c** ώστε η πρώτη διεργασία:

- αρχικά να ζητάει πόσα στοιχεία θα διαβάσει από το χρήστη,
- θα διαβάσει πραγματικούς αριθμούς από το πληκτρολόγιο τόσα σε πλήθος όσα έχει δηλώσει ο χρήστης στο προηγούμενο ερώτημα,
- θα στέλνει τα δεδομένα στις υπόλοιπες διεργασίες όπως παραπάνω.
- Να δοθεί ο πηγαίος κώδικας, ως **c6.c .**