ΠΑΝΕΠΙΣΤΗΜΙΟ
ΔΥΤΙΚΗΣ ΜΑΚΕΔΟΝΙΑΣ

Συστήματα Παράλληλης και Κατανεμημένης Επεξεργασίας

Ενότητα: ΕΡΓΑΣΤΗΡΙΑΚΗ ΑΣΚΗΣΗ **MTL (Manycore Testing Laboratory)**

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.

Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

Περιεχόμενα

Σκοπός της άσκησης

- Εξοικείωση και χρήση του πολυπύρηνου συστήματος Intel Many Core (MTL), και των προγραμμάτων που βρίσκονται εγκατεστημένα.

Τα παραδοτέα cxx-xx είναι συνήθως screenshots ή απαντήσεις κειμένου που δείχνουν την επιτυχία της συγκεκριμένης ενέργειας.

1. Σύνδεση και Compile hello world

- Συνδεθείτε στο MTL, μέσω των οδηγιών που έχουν δοθεί από το διδάσκοντα.
- Κατασκευάστε το πρόγραμμα **helloworld.c** στον κατάλογο εργασίας σας.
- Κάντε compile to helloworld.c με το gcc. Δε θα πρέπει να εμφανίζεται κανένα warning ή error.
- **(C01-1)** Εκτελέστε το αρχείο που έχει δημιουργηθεί, και επιβεβαιώστε την εμφάνιση του μηνύματος "`Hello World from MTL`".

2. Σύνδεση μέσω γραφικής διεπαφής μονού παραθύρου

Προκειμένου να εμφανίσουμε στην οθόνη μας, εφαρμογές που απαιτούν γραφικά σε X Windows (όπως π.χ. ένα browser), θα πρέπει εκτός από την ασφαλή σύνδεση με SSH, να εκτελέσουμε στο μηχάνημά μας έναν X-Server σε κατάσταση ακρόασης *(δηλαδή, να δέχεται συνδέσεις)*, να ενεργοποιήσουμε το X11Forwarding στο SSH, και να ρυθμίσουμε κατάλληλα τη μεταβλητή φλοιού DISPLAY. Αυτό μπορεί να γίνει με πολλούς τρόπους. Η παρακάτω διαδικασία είναι η πιο απλή, και συνίσταται για τα άτομα που δεν έχουν σχετική εμπειρία.

- Σε περίπτωση που χρησιμοποιούμε Microsoft Windows:
  - Κατεβάζετε στον υπολογιστή σας το πρόγραμμα MobaXterm το οποίο είναι ταυτόχρονα και πρόγραμμα ασφαλής σύνδεσης SSH και πρόγραμμα εμφάνισης X11 γραφικών. *(:/ /mobaxterm.mobatek.net/download-home-edition.html , προτιμήστε την έκδοση portable που δε χρειάζεται εγκατάσταση).*
  - Μόλις το εκτελέσετε, ενδέχεται να ερωτηθείτε σχετικά με το να ανοίξει μια θύρα του firewall για τον X11 Server. Αποδεχτείτε το.
- Σε περίπτωση που χρησιμοποιούμε Linux, δε χρειάζεται κάτι ιδιαίτερο.

- Στην προτροπή του προγράμματος ή του τερματικού, δώστε:
  `ssh -Y <user>@zafora.icte.uowm.gr` για να συνδεθείτε με το zafora, το οποίο έχει άμεση πρόσβαση στο MTL, με την παράμετρο **`-Y`** που υποδηλώνει αυτόματη προώθηση X11 Display,
  (ΠΡΟΣΟΧΗ: Η παράμετρος -Y τοποθετείται ΚΑΘΕ φορά που δίνετε την εντολή ssh. Από το τοπικό μας μηχάνημα για να συνδεθούμε στο zafora και από το zafora για να συνδεθούμε στο pleiades).
  π.χ. **`ssh -Y mdasyg@zafora.icte.uowm.gr`**
- Μόλις συνδεθείτε, επιβεβαιώστε ότι έχετε αυτόματη προώθηση με το να δείτε την τιμή της μεταβλητής **$DISPLAY**

  ```
  echo $DISPLAY
  ```

  και θα δείτε κάτι παρόμοιο με τη γραμμή: **`localhost:10.0`** . Αν είναι κενή η έξοδος της εντολής, τότε έχετε ξεχάσει να βάλετε την παράμετρο **`-Y`** (σε κάποια εντολή ssh, που μπορεί να είναι και η αρχική).
  - Δοκιμάστε την προώθηση γραφικών, με το να δώσετε **`xterm`** στο zafora. Μετά από 2 δευτερόλεπτα θα δείτε να ανοίγει ένα νέο παράθυρο στο σύστημά σας με τίτλο xterm.
- Πατήστε exit στο xterm, γιατί δε μας ενδιαφέρουν τα γραφικά στο zafora, αλλά στο MTL (δηλαδή στο pleides.icte.uowm.gr).
- Από το zafora συνδεθείτε στο MTL με το ssh και την παράμετρο **-Y** για την ενεργοποίηση της αυτόματης προώθησης X11. Η εντολή θα είναι παρόμοια: **`ssh -Y pleiades`**
- **(C02-1)** Στην προτροπή, δώστε xterm. Αν ανοίξει το νέο παράθυρο φλοιού μετά από 2 λεπτά, με τίτλο xterm, μπορείτε να χρησιμοποιήσετε χωρίς κανένα πρόβλημα, εφαρμογές γραφικών που βρίσκονται στο MTL.

3. Σύνδεση μέσω γραφικής διεπαφής διακομιστή

Ο επόμενος τρόπος χρησιμοποιεί το VNC στο MTL. Προκειμένου να συνδεθούμε σε αυτό, θα πρέπει να δημιουργήσουμε μια αλυσίδα προώθησης θυρών, αν συνδέεστε εκτός του UOWM.

- Συνδεθείτε στο MTL μέσω SSH.
  - **(Μόνο την πρώτη φορά)** Εκτελέστε το πρόγραμμα **`vncserver`**. Μόλις το εκτελέσετε θα δώσετε έναν κωδικό *(μόνο την πρώτη φορά)*, και στη συνέχεια θα σας αναφερθεί η θύρα ακρόασης *(π.χ.* **`New 'acano01:2 (msd)' desktop is acano01:2`***)*, δηλαδή στο παράδειγμά μας είναι η XVnc θύρα με νούμερο 2, και επειδή οι θύρες Xvnc ξεκινάνε από τον αριθμό 59**XX**, είναι η 5902 *(επιβεβαιώστε με* **`netstat -an | grep 5902`** *, και θα δείτε ότι υπάρχει ένα LISTEN socket).* Επίσης, μπορείτε να διαπιστώσετε, ότι αυτόματα ανοίγει η Xvnc HTTP θύρα που ξεκινάει από το 58**XX**, δηλαδή η 5802 και η X11 θύρα που ξεκινάει από το 60**XX**, δηλαδή η 6002.
  - Τις επόμενες φορές δε χρειάζεται να εκτελέσετε το **`vncserver`**, εκτός αν γίνει reboot στο MTL ή το κλείσετε εσείς. Μπορείτε με
- **`ps xuw`** να δείτε αν εκτελείται η δικιά σας διεργασία Xvnc στο MTL.

  ΠΡΟΣΟΧΗ: Μόνο μια διεργασία Xvnc επιτρέπεται να εκτελείται στο MTL.
- **ΠΡΟΣΟΧΗ: Σε περίπτωση που εκτελέσετε το vncserver και δε σας δώσει αριθμό, αλλά εμφανίσει μια σειρά από μηνύματα της μορφής:**

  ```
  Warning: pleiades.icte.uowm.gr:XXX is taken because of /tmp/.XXX-lock
  ```

  και τελικό μήνυμα:

  ```
  vncserver: no free display number on pleiades.icte.uowm.gr.
  ```

  Αυτό σημαίνει ότι οι θύρες προεπιλογής έχουν καταληφθεί από συμφοιτητές σας και τότε θα πρέπει να χρησιμοποιήσετε μια επιπρόσθετη παράμετρο μετά την εντολή vncserver για να λάβετε μια γραφική οθόνη σε διαφορετικό εύρος. Η παράμετρος είναι άνω-κάτω τελεία ΚΑΙ ένας τυχαίος αριθμός που ανήκει στο εύρος 20000 έως 30000. Π.χ.

  `vncserver :24XXX`

  Μόλις δώσετε αυτή την εντολή (και δεν είναι κατειλημμένη αυτή η θύρα, θα εμφανιστεί ένα μήνυμα παρόμοιο με:

  ```
  New 'pleiades.icte.uowm.gr:24XXXX (mdasygenis)' desktop is pleiades.icte.uowm.gr:XXXX
  Starting applications specified in /zstorage/home/mdasygenis/.vnc/xstartup
  Log file is /zstorage/home/mdasygenis/.vnc/pleiades.icte.uowm.gr:24XXX.log
  ```

  για να βρείτε τις ακριβές θύρες που σας έχουν αποδοθεί, θα πρέπει να διαβάσετε το log file με την εντολή:

  `more ~/.vnc/pleiades.icte.uowm.gr:24XXX.log`

  όπου θα βρείτε τις γραμμές:

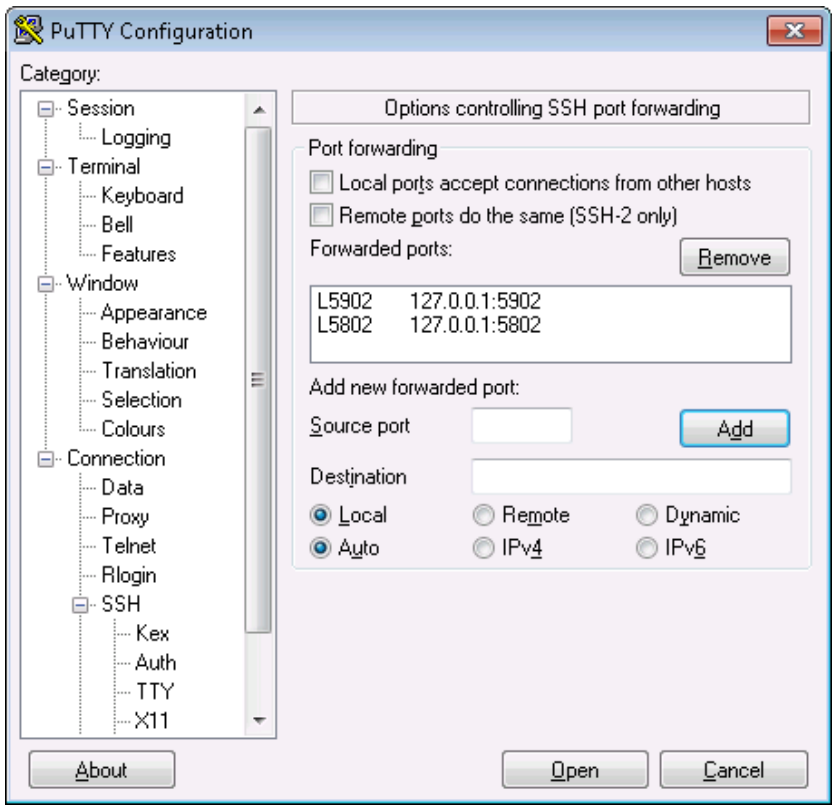
  ```
  Thu Mar 2 11:53:12 2017
  vncext: VNC extension running!
  vncext: Listening for VNC connections on port 26YYY
  vncext: Listening for HTTP connections on port 26ZZZ
  vncext: created VNC server for screen 0
  ```

δηλαδή ότι η σύνδεση μέσω vncviewer μπορεί να γίνει στο
`pleiades.icte.uowm.gr:26YYY`
ενώ η σύνδεση μέσω webbroser μπορεί να γίνει στο
`http://pleiades.icte.uowm.gr:26ZZZZ`

➔Οι οδηγίες είναι διαφορετικές για το αν βρίσκετε ΕΝΤΟΣ ή ΕΚΤΟΣ του δικτύου του uowm. Επιλέξτε να ακολουθήσετε τις σωστές οδηγίες.

- Βγείτε από το MTL, γιατί δε χρειάζεται άλλο η συγκεκριμένη SSH σύνδεση.
- Αν είστε ΕΝΤΟΣ του Πανεπιστημίου ή χρησιμοποιείτε VPN, δε χρειάζεται κάποια προώθηση θύρας αφού το τείχος προστασίας ΔΕ θα σας εμποδίσει.

- Αν είστε ΕΚΤΟΣ του Πανεπιστημίου τότε θα πρέπει να χρησιμοποιήσετε προωθήσεις θυρών για να συνδεθείτε στη γραφική διεπαφή μέσω του τείχους προστασίας:
  - Στον υπολογιστή σας, αν το ΛΣ είναι τύπου Unix, χρησιμοποιήστε το ssh με την παράμετρο **`-L`** για την προώθηση των θυρών, όπως περιγράφεται παρακάτω.
  - Στον υπολογιστή σας, αν το ΛΣ είναι Microsoft Windows, ανοίξτε το putty, και στις ρυθμίσεις **Connection->SSH->Tunnels**, δώστε στο source port τον αριθμό που έχετε σημειώσει πιο πριν και στο destination port τον αριθμό 127.0.0.1 και αυτό που είχατε σημειώσει πιο πριν με άνω-κάτω τελεία ως τη Xvnc θύρα, π.χ. 127.0.0.1:5902 . Πατήστε ADD και θα προστεθεί στη λίστα. Επίσης, προσθέστε και την HTTP θύρα, π.χ. 127.0.0.1:5802 , την οποία θα έχετε συνδέσει με την αντίστοιχη source port, π.χ. 5802. Μη ξεχάσετε να πατήσετε Add. Πρέπει να εμφανίζονται 2 καταχωρήσεις στο **Forwarded Ports**.



- Στο putty, στο session tab δώστε τα στοιχεία σύνδεσης με το zafora, και συνδεθείτε με το σύστημα.
- Από το zafora, ενεργοποιήστε μια ssh σύνδεση προς το MTL με το πρόγραμμα SSH και την παράμετρο προώθησης **-L** *(δυο φορές)* για την προώθηση θυρών, με παρόμοια σύνταξη όπως παρακάτω:

```
ssh -L 127.0.0.1:5902:127.0.0.1:5902 -L 127.0.0.1:5802:127.0.0.1:5802 mtl_hostname
```

με κατάλληλη τροποποίηση των στοιχείων.

(ΜΟΝΟ Σε περίπτωση που είστε ΕΚΤΟΣ του Πανεπιστημίου και δε χρησιμοποιείτε UOWM VPN.)

Συνδεθείτε με επιτυχία στο MTL, και αφήστε το παράθυρο ανοιχτό. Από τον τοπικό σας υπολογιστή:

- **(C03-1)** Δοκιμάστε την HTTP σύνδεση με το να ανοίξετε ένα φυλλομετρητή, και να δώσετε *(με κατάλληλη τροποποίηση της HTTP θύρας)*. Αφού δώσετε τον κωδικό σας θα δείτε τον X11 διακομιστή στο MTL.
    - **(C03-1)** Δοκιμάστε την Xvnc σύνδεση με το να χρησιμοποιήσετε ένα πρόγραμμα πελάτη Xvnc *(π.χ. για Microsoft Windows , ενώ για Linux το* `vncviewer` *είναι ηδη εγκατεστημένο)* και να συνδεθείτε στη διεύθυνση 127.0.0.1:5902 *(με κατάλληλη τροποποίηση της XVnc θύρας)*. Αφού δώσετε τον κωδικό σας θα δείτε τον X11 διακομιστή στο MTL.

Η υποστήριξη για το NAPI JAVA PLUGIN έχει καταργηθεί στους περισσότερους browsers. Αγνοήστε το βήμα της http σύνδεσης.

(ΜΟΝΟ Σε περίπτωση που είστε ΕΝΤΟΣ του Πανεπιστήμιου ή χρησιμοποιείτε UOWM VPN)

- **(C03-1)** Δοκιμάστε την HTTP σύνδεση με το να ανοίξετε ένα φυλλομετρητή (κατά προτίμηση το Firefox), και να δώσετε *(με κατάλληλη τροποποίηση της HTTP θύρας)* `http://pleiades.icte.uowm.gr:59XX` . Αφού δώσετε τον κωδικό σας θα δείτε τον X11 διακομιστή στο MTL. Σε περίπτωση που σας εμφανίζει κάποιο κενό παράθυρο τότε δεν έχει εγκατασταθεί το κατάλληλο java plugin στο φυλλομετρητή ή δεν είναι ενεργοποιημένη η Java ή εκτελείτε Chrome που δε δέχεται τα Java plugins του συγκεκριμένου τύπου.
- **(C03-1)** Δοκιμάστε την Xvnc σύνδεση με το να χρησιμοποιήσετε ένα πρόγραμμα πελάτη Xvnc *(π.χ. για Microsoft Windows , ενώ για Linux το* `vncviewer` *είναι ήδη εγκατεστημένο)* και να συνδεθείτε στη διεύθυνση `pleiades.icte.uowm.gr:59XX` *(με κατάλληλη τροποποίηση της XVnc θύρας)*. Αφού δώσετε τον κωδικό σας θα δείτε τον X11 διακομιστή στο MTL.

4. Κατανόηση του συστήματος

Ο καλός προγραμματιστής πρέπει να γνωρίζει την αρχιτεκτονική στην οποία εργάζεται. Για να βελτιστοποιηθεί μια εφαρμογή θα πρέπει να ληφθούν κάποιες αποφάσεις, οι οποίες εξαρτώνται από την αρχιτεκτονική. Συγκεκριμένα, πρέπει να γνωρίζει τι επεξεργαστές έχει, πόσα επίπεδα κρυφής μνήμης και με τι χωρητικότητα το κάθε επίπεδο, ποιες κρυφές μνήμες διαμοιράζονται σε ποιους επεξεργαστικούς πυρήνες κτλ. Σε αυτή την εργασία θα χαρτογραφήσετε το σύστημα MTL.

Ο πιο εύκολος τρόπος για να το κάνετε αυτό είναι να χρησιμοποιήσετε τη σουίτα προγραμμάτων likwid (). Αν το πρόγραμμα είναι ήδη εγκατεστημένο μπορείτε να προσπεράσετε τα παρακάτω βήματα. Δείτε αν υπάρχει στο MTL στον κατάλογο **`/opt/likwid`** . Το εκτελέσιμο βρίσκεται στον υποκατάλογο bin , δηλαδή έχει πληρη διαδρομή **`/opt/likwid/bin`** .

ΑΝ ΔΕΝ ΥΠΑΡΧΕΙ ΤΟ /opt/likwid θα πρέπει να το εγκαταστήσετε ως εξής:

- Συνδεθείτε μέσω τερματικού με το MTL.
- Δημιουργήστε στον κατάλογο εργασίας σας, έναν υποκατάλογο tools.
- Μεταφορτώστε με όποιον τρόπο θέλετε το αρχείο
- likwid-3.0.0.tar.gz ή νεότερη έκδοση από τη σελίδα του project likwid *(π.χ. με WinSCP, scp , wget , ftp κτλ).*
- Αποσυμπιέστε το αρχείο που μόλις κατεβάσατε με τα εργαλεία gunzip και tar.
- Εισέλθετε στον κατάλογο που έχει δημιουργηθεί.
- Κάντε make για να δημιουργηθούν τα εκτελέσιμα για την αρχιτεκτονική μας. Μόλις ολοκληρωθεί η διαδικασία μπορείτε να το εκτελέσετε:

- Μεταβείτε στον κατάλογο που βρίσκεται το αρχείο **likwid-topology**. **(π.χ. `cd /opt/likwid/bin`)**
- Εκτελέστε το **`./likwid-topology`** και **`./likwid-topology -C`** και **`cat /proc/cpuinfo`**
- Βρείτε τις παρακάτω απαντήσεις:
  - **(C04-1)** Ποιος είναι ο τύπος *(=όνομα)* του επεξεργαστή;
  - **(C04-2)** Σε τι συχνότητα ρολογιού είναι χρονισμένος;
  - **(C04-3)** Πόσες ξεχωριστές θέσεις για επεξεργαστές *(sockets)* έχει το σύστημά;
  - **(C04-4)** Πόσοι πυρήνες βρίσκονται σε κάθε socket *(στο ίδιο chip);*
  - **(C04-5)** Ο επεξεργαστής υποστηρίζει υπερνημάτωση *(δηλαδή ξεχωριστά hardware threads)* ;
  - **(C04-6)** Πόσοι είναι συνολικά οι πραγματικοί επεξεργαστικοί πυρήνες, και πόσοι οι πραγματικοί + εικονικοί ;
  - **(C04-7)** Πόσα επίπεδα κρυφής μνήμης έχει το σύστημα ;
  - **(C04-8)** Μια κρυφή μνήμη επιπέδου 1 διαμοιράζεται με άλλους πραγματικούς επεξεργαστές; Δικαιολογήστε την απάντηση.

- **(C04-9)** Μια κρυφή μνήμη επιπέδου 2 διαμοιράζεται με άλλους πραγματικούς επεξεργαστές; Δικαιολογήστε την απάντηση.
- **(C04-10)** Μια κρυφή μνήμη επιπέδου 3 διαμοιράζεται με άλλους πραγματικούς επεξεργαστές; Δικαιολογήστε την απάντηση.
- **(C04-11)** Πόση είναι η συνολική off-chip μνήμη RAM του συστήματος;
- **(C04-12)** Πόσες είναι οι περιοχές NUMA της μνήμης;

Εκτελέστε το `./likwid-topology -g` και δείτε με γραφική αναπαράσταση την ιεραρχία της μνήμης για να επιβεβαιώσετε τις απαντήσεις σας.

5. Intel Vtune Amplifier 2013 για εύρεση των κρίσιμων σημείων

Εξοικείωση με το εργαλείο Intel Vtune Amplifier 2013 για την εύρεση των κρίσιμων σημείων *(hotspots).*

Σε αυτή την εργασία θα χρησιμοποιήσουμε το εργαλείο Intel Vtune Amplifier 2013 της Intel, προκειμένου να αναλύσουμε ένα κώδικα, να βρούμε τα κρίσιμα σημεία κώδικα, δηλαδή τα σημεία που απαιτούν συσσωρευτικά τους περισσότερους κύκλους, να βελτιστοποιήσουμε τον κώδικα σε αυτά τα σημεία, και να διαπιστώσουμε την επίτευξη των στόχων μας.

Θα χρησιμοποιήσουμε τις αναλυτικές οδηγίες που αναγράφονται στο έγγραφο "`hotspots_amplxe_lin.pdf`". Θα πρέπει να προσέξετε στη σύνταξη των εντολών, γιατί αν κάνετε κάποιο λάθος και αργήσετε να το παρατηρήσετε, θα πρέπει να γυρίσετε κάποιες σελίδες πίσω για να επαναλάβετε την άσκηση με τις σωστές εντολές.

Με το πέρας της άσκησης θα πρέπει να μπορείτε να απαντήσετε στις εξής ερωτήσεις:

- **(C05-1) Ποια είναι τα πρώτα βήματα για να αναλύσουμε την εφαρμογή μας, πριν εκτελέσουμε το amplifier και αμέσως μόλις το εκτελέσουμε;**
- **(C05-2) Πως βρίσκουμε τα hotspots με το amplifier;**
- **(C05-3) Πως μπορούμε άμεσα να τροποποιήσουμε τον κώδικα μέσα από το πρόγραμμα amplifier;**

5.1 Διαδικασία Εκτέλεσης Άσκησης

- Συνδεθείτε μέσω της γραφικής διεπαφής VNC στο MTL, όπως έχετε κάνει σε προηγούμενο εργαστήριο.
- Για να χρησιμοποιήσετε το εργαλείο Intel Vtune Amplifier, θα πρέπει να αρχικοποιήσετε το περιβάλλον *(δηλαδή, να τεθούν κάποιες μεταβλητές, όπως και να τροποποιηθεί η διαδρομή αναζήτησης των εκτελέσιμων εφαρμογών για να προστεθεί η τοποθεσία του amplifier).* Το πρόγραμμα έχει εγκατασταθεί στον κατάλογο /opt/intel/vtune_amplifier_xe_2013/. Μέσα στον κατάλογο βρίσκεται το αρχείο αρχικοποίησης amplxe-vars.sh. Δώστε
  `source /opt/intel/vtune_amplifier_xe_2013/amplxe-vars.sh`
  (Μπορείτε να δείτε τα περιεχόμενα του με `more`*)*
- Στο συγκεκριμένο tutorial θα χρησιμοποιήσουμε τον κώδικα tachyon που βρίσκεται συμπιεσμένος στο αρχείο: `tachyon_vtune_amp_xe.tgz` μέσα στη διαδρομή `/opt/intel/vtune_amplifier_xe_2013/samples/en/C++/` Αποσυμπιέστε το αρχείο αυτό στον κατάλογο εργασίας *(η εντολή εκτελείται σε μια γραμμή)*:
  `cd ; tar -xvzf/opt/intel/vtune_amplifier_xe_2013/samples/en/C++/tachyon_vtune_amp_xe.tgz`
- Εισέλθετε στον κατάλογο tachyon που έχει δημιουργηθεί και δώστε την εντολή συμβολομετάφρασης του project:
  `cd tachyon ; make`
  Σε περίπτωση που εμφανιστεί πρόβλημα στο compilation, θα πρέπει να κάνετε τα εξής βήματα:
  (α) επεξεργάζεστε το αρχείο ~/tachyon/Makefile
  (b) βρίσκετε τη γραμμή που ξεκινάει με LIBS
  (c) προσθέτετε στο τέλος την παράμετρο `-lpthread`
- Συνεχίστε το tutorial, σημειώνοντας τις απαντήσεις στις ερωτήσεις **με επισήμανση**.
- **(C05-4) Ποιος compiler χρησιμοποιήθηκε για τον κώδικα; Ο gcc ή ο icc;**
- **(C05-5) Κατά την εκτέλεση του μη βελτιστοποιημένου κώδικα, πόσος είναι ο συνολικός χρόνος CPU;**

***** Όλες οι παραπάνω απαντήσεις βρίσκονται έως και τη σελίδα 13 *****

- Προσοχή: Κατά τη δημιουργία του νέου project να τοποθετήσετε το σωστό working directory, όπως και τη σωστή παράμετρο, διαφορετικά δε θα μπορεί να εκτελεστεί η εφαρμογή *(και θα εμφανιστεί το μήνυμα no data ή κάτι παρόμοιο κατά την εκτέλεση της εφαρμογής)*.
- **(C05-6) Το εργαλείο που χρησιμοποιείτε σε αυτό το εργαστήριο ποιες αναλύσεις υποστηρίζει αλγοριθμικού τύπου *(algorithm analysis)*;**
- **(C05-7) Κατά την εκτέλεση του μη βελτιστοποιημένου κώδικα στο εργαλείο amplifier, πόσος είναι ο συνολικός χρόνος CPU; Είναι μεγαλύτερος μικρότερος ή ίδιος με τον προηγούμενο συνολικό χρόνο εκτέλεσης;**
- **(C05-8) Η μη βελτιστοποιημένη έκδοση πόσα νήματα χρησιμοποιεί; Πως το βρήκατε;**

- **(C05-9) Ποιοι είναι οι χρόνοι CPU Time που μετρήσατε στο σύστημά σας, για τις τρεις πιο χρονοβόρες συναρτήσεις;**
- Προς το τέλος του summary υπάρχει το CPU Usage Histogram. Το πρόγραμμα έχει τοποθετήσει σε μια τιμή του οριζόντιου άξονα *(simultaneously utilized logical cpus)*, μια κατακόρυφη γραμμή με όνομα "Target Concurrency". **(C05-10) Ποια είναι αυτή η τιμή και πως έχει υπολογιστεί αυτόματα για το σύστημά μας;**
- **(C05-11) Στο παράθυρο bottom-up, ποια συνάρτηση κάλεσε το initialize_2D_buffer;**
- **(C05-12) Στο παράθυρο bottom-up, ποια συνάρτηση κάλεσε το sphere_intersect;**
- **(C05-13) Στο παράθυρο bottom-up, στα δεξία υπάρχει το call stack. Ποια συνάρτηση κάλεσε την initialize_2D_buffer, σε ποια αρχείο και σε ποια γραμμή βρίσκεται;**

***** Όλες οι παραπάνω απαντήσεις βρίσκονται έως και τη σελίδα 17 *****

- Στο timeline στο παράθυρο bottom-up, αν αφήσουμε το mouse πάνω από το γράφημα, θα μας αναφερθεί ότι το utilization είναι περίπου 98% με 100%. Εντούτοις, το πρόγραμμα αναφέρει ότι το πρόγραμμα έχει "poor" cpu utilization. **(C05-14) Γιατί γίνεται αυτό; Ποια είναι η μέγιστη χρήση επί τις % για το σύστημα MTL;**
- Αφού πατήσετε διπλό κλικ στη συνάρτηση, θα εμφανιστεί ο κώδικας. Πατήστε το εικονίδιο που θα σας μεταφέρει στη γραμμή της συνάρτησης με τον περισσότερο CPU Time *(Go to Biggest Function Hotspot)*.
- Ενεργοποιήστε την εμφάνιση της assembly, πατώντας το αντίστοιχο κουμπί. **(C05-15) Η γραμμή που έχετε επιλέξει *(με τη μεγαλύτερη επιβάρυνση)* από πόσες εντολές assembly αποτελείται; Από τις εντολές assembly, ποια είναι η εντολή με τον περισσότερο χρόνο εκτέλεσης;**
- Στη σελίδα 22 γίνεται μια αλλαγή στον κώδικα με το να τοποθετήσετε σε σχόλια μια συνάρτηση, και να απομακρύνετε τα σχόλια από μια άλλη.
  (C05-16) Ποιο σκοπό επιτελούν αυτές οι δυο συναρτήσεις και γιατί η πρώτη συνάρτηση απαιτεί περισσότερους κύκλους CPU από τη δεύτερη;

***** Όλες οι παραπάνω απαντήσεις βρίσκονται έως και τη σελίδα 22 *****

- **(C05-17) Μετά την τροποποίηση του κώδικα, και την επαναδημιουργία του εκτελέσιμου αρχείου, πόσος χρόνος CPU απαιτείται; Έχει βελτιωθεί ο χρόνος;**
- Από το "Analysis Type" πατήστε Re-resolve για να εκτελεστεί ο νέος κώδικας, και στη συνέχεια το εικονίδιο της σύγκρισης των αποτελεσμάτων.
- **(C05-18) Στο summary της σύγκρισης των αποτελεσμάτων, πόσο αναφέρεται ότι έχει μειωθεί το CPU Time;**
- **(C05-19) Στο summary η συνάρτηση initialize_2D_buffer πόσο έχει μειωθεί σε CPU Time;**
- **(C05-20) Στο bottom-up της σύγκρισης *(grouping: function /call stack)*, να βρείτε τις 3 συναρτήσεις που έχουν τη μεγαλύτερη διαφορά *(βελτίωση χρόνου)* στο CPU Time: Difference by Utilization.**

***** Όλες οι παραπάνω απαντήσεις βρίσκονται έως και τη σελίδα 26 *****

5.2 Χρήση διαφορετικού compiler και σύγκριση

Μπορείτε να χρησιμοποιήσετε διαφορετικό compiler αν ενεργοποιήσετε τις κατάλληλες μεταβλητές περιβάλλοντος. Στο MTL υπάρχει εγκατεστημένος και ο compiler Intel C Compiler. Καλείστε να χρησιμοποιήσετε αυτό το compiler.

Αρχικοποιήστε το περιβάλλον του Intel Compiler με το να δώσετε:
`source /opt/intel/Compiler/latest/bin/iccvars.sh intel64`
Επιβεβαιώστε ότι το icc βρίσκεται στη διαδρομή αναζήτησης με το να δώσετε:
`icc -v`
και να δείτε ότι σας εμφανίζει σχετικό μήνυμα.

Απομακρύνετε τα παλαιά εκτελέσιμα αρχεία και επαναλάβετε τη διαδικασία compilation (`make clean ; make`).

** **Προσοχή****: Μερικές φορές απαιτούνται έξτρα header files που δε βρίσκονται στο include path του Intel Compiiler. Θα πρέπει να προστεθούν αυτά τα header files στο include path. Αρχικά βρίσκετε την τοποθεσία του header file με την εντολή locate. Έστω για παράδειγμα, δεν υπάρχει το αρχείο bits/c++config.h και στην εντολή
`#include <bits/c++config.h>`
Εμφανίζεται τα μήνυμα:

```
/usr/include/c++/7/cassert(43): catastrophic error: cannot
open source file "bits/c++config.h"
```

Τότε βρίσκουμε το αρχείο με την εντολή `locate c++config.h` .

Μας εμφανίζονται διάφορα path. Η εύρεση του σωστού αρχείου απαιτεί εξοικείωση με το Linux και τη συμβολομετάφραση. Επιλέγουμε τη διαδρομή που ξεκινάει με `/usr` και αφορά έκδοση header file συμβατή με την έκδοση του intel compiler που είναι εγκατεστημένο στο σύστημα (το 2018 η συμβατή έκδοση είναι η 4.8). Π.χ. φαίνεται ότι το `c++config.h` βρίσκεται σε διάφορες διαδρομές:

….

```
/usr/include/x86_64-linux-gnu/c++/4.8/32/bits/c++config.h
/usr/include/x86_64-linux-gnu/c++/4.8/bits/c++config.h
/usr/include/x86_64-linux-gnu/c++/4.8/x32/bits/c++config.h
/usr/include/x86_64-linux-gnu/c++/7/32/bits/c++config.h
/usr/include/x86_64-linux-gnu/c++/7/bits/c++config.h
/usr/include/x86_64-linux-gnu/c++/7/x32/bits/c++config.h
/usr/lib/x86_64-linux-gnu/sigc++-2.0/include/sigc++config.h
```

…

Επιλέγουμε μια διαδρομή που έχει το 4.8 και επίσης δεν έχει το 32 (γιατί δε θέλουμε 32bit έκδοση). Η σωστή επιλογή είναι η:

```
/usr/include/x86_64-linux-gnu/c++/4.8/bits/c++config.h
```

Τοποθετούμε το path /usr/include/x86_64-linux-gnu/c++/4.8/ στο header include path ως εξής (είναι μια γραμμή):

```
export CPLUS_INCLUDE_PATH=/usr/include/x86_64-linux-gnu/c++/4.8:$CPLUS_INCLUDE_PATH
```

Μπορούμε να επαναλάβουμε την ανωτέρω εντολή με άλλο path, για κάθε header file (συνεχώς κάνει prepend ένα νέο include path χωρίς να διαγράφει τα προηγούμενα).

Η εντολή export ισχύει ΜΟΝΟ για το τρέχων τερματικό παράθυρο, και αν κλείσει ή κάνουμε logout, τότε σταματάει να ισχύει. Αν θέλετε να γίνει μόνιμη, θα πρέπει να την προσθέσετε στο αρχείο αρχικοποίησης του τερματικού σας παραθύρου για το φλοιό που χρησιμοποιείτε, π.χ. για το bash είναι το `$HOME/.profile` .

Εκτελέστε την εφαρμογή με την κατάλληλη είσοδο, όπως στην αρχή της άσκησης και συγκρίνετε το χρόνο ως προς το compilation με το g++.

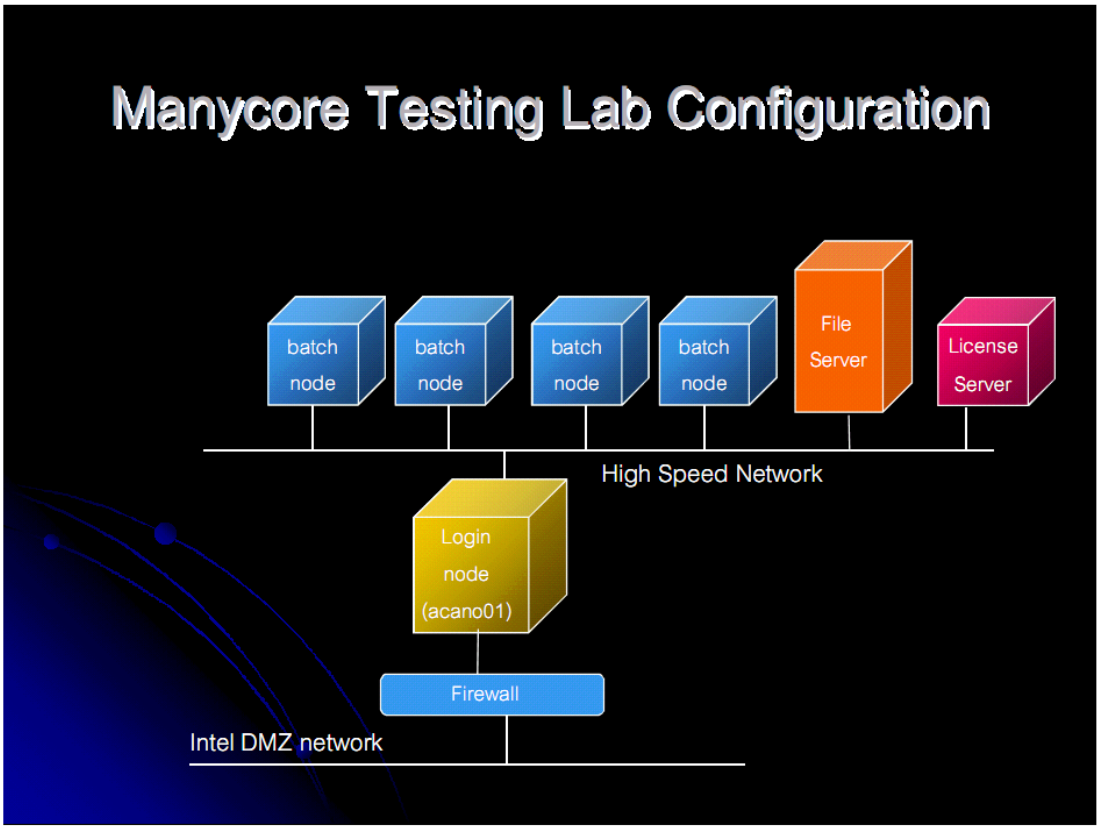
(C05-21) Υπάρχει διαφορά; Ποιος compiler δημιούργησε πιο βελτιωμένο εκτελέσιμο;

Πηγαίνετε στο Intel Vtune Amplifier => Analysis Type και πατήστε Re-resolve. Στη συνέχεια πατήστε το κουμπί της σύγκρισης των αποτελεσμάτων *(summary)*.

(C05-22) Τέλος, παρατηρήστε στο Bottom-up σε ποια συνάρτηση υπάρχει η καλύτερη βελτίωση.

6. Σύστημα υποβολής εργασιών PBS στο MTL

Ως τώρα έχετε χρησιμοποιήσει μόνο τον κεντρικό υπολογιστή του MTL *(που είναι και ο login node)*. Η αρχιτεκτονική του MTL όμως, δεν αποτελείται μόνο από αυτόν τον κόμβο. Όπως φαίνεται στο παρακάτω σχήμα, εκτός από τον login node, υπάρχουν πολλαπλοί κόμβοι (=κόμβοι εργάτες) που συνδέονται με δίκτυο διασύνδεσης υψηλής ταχύτητας *(Gigabit ή Fiber Optic),* και αναλαμβάνουν εργασίες από την ουρά υποβολής εργασιών.

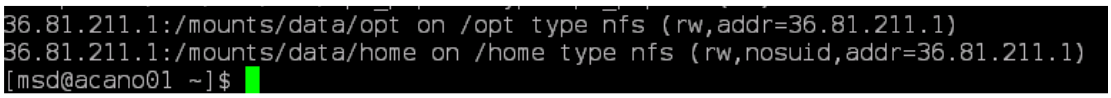


Οι κόμβοι εργάτες δεν επιτρέπουν τη σύνδεση χρηστών αλλά είναι μόνο batch nodes, που σημαίνει ότι λειτουργούν σε κατάσταση slave και δέχονται εντολές ή προγράμματα προς εκτέλεση από το login node. Το login node αποφασίζει ποια εργασία θα εκτελεστεί σε ποιον εργάτη. Για αυτό άλλες φορές το πρόγραμμα μας θα εκτελείται σε έναν κόμβο εργάτη01, άλλες φορές σε έναν κόμβο εργάτη02 κ.ο.κ.

Η διαδικασία εκτέλεσης ενός προγράμματος στο MTL, στο οποίο θέλουμε να συμμετέχουν επεξεργαστικά και οι batch node, είναι η εξής:

1. Δημιουργία της εφαρμογής μας στο login node, έλεγχος και επιβεβαίωση ορθής λειτουργίας (=εκτέλεση μιας φοράς στο login node).
2. Δημιουργία αρχείου υποβολής εργασίας, στο οποίο αναγράφονται οι επεξεργαστές που ζητούνται, η μνήμη και άλλα στοιχεία.
3. Υποβολή της εργασίας στο MTL με το εργαλείο **`qsub`**.
4. Επίβλεψη της εκτέλεσης με το εργαλείο **`qstat`**.
5. Συλλογή των αποτελεσμάτων.

Το εργαλείο qsub είναι ένα εργαλείο τερματικού που επιτρέπει την υποβολή εργασιών σε μια σειρά από batch nodes, που συνεργάζονται ως συστοιχία PBS[1]. Τα batch nodes είναι κόμβοι με πολλαπλούς πυρήνες, όπου συνδέονται σε ένα κοινό σύστημα αρχείων NFS, όπως φαίνεται από την εικόνα *(για τον κατάλογο /home):*



Από τη σελίδα βοήθειας του **qsub** να απαντήσετε:

(C06-1) Μπορούν να υποβληθούν διαλογικές εργασίες *(interactive jobs)* με το qsub;

(C06-2) Ποια είναι η παράμετρος στο qsub που προσδιορίζει ποια χρονική στιγμή θα υποβληθεί η εργασία;

(C06-3) Η παράμετρος -l τι προσδιορίζει στο qsub;

Από τη σελίδα βοήθειας του **pbs_resources** να απαντήσετε:

(C06-4) Ο περιορισμός walltime τι προσδιορίζει;

(C06-5) Ο περιορισμός ompthreads τι προσδιορίζει;

Κατά τη σύνδεση στο login node, καθορίζεται η μεταβλητή περιβάλλοντος **$PBS_SERVER** όπου καθορίζει τον υπολογιστή χρονοδρομολόγησης των batch εργασιών, και ο οποίος είναι αυτός που θα δεχτεί την εργασία από το **qsub**.

(C06-6) Τι τιμή έχει αυτή η μεταβλητή και πως βρήκατε τα περιεχόμενα της; Αν είναι κενή αναφέρετε ότι είναι κενή.

Το **qsub** υποβάλει μόνο αρχεία φλοιού. Δημιουργήστε το αρχείο φλοιού **c1.sh**, στο οποίο εκτυπώνετε κάποιες βασικές πληροφορίες για τον κόμβο στον οποίο θα εκτελεστεί (μη ξεχάσετε **chmod 750 ./c1.sh**)

```
#!/bin/sh
echo "Working directory is $PWD, at `hostname`"
echo "I see disks: `mount`"
```

Επιβεβαιώστε την εκτέλεση στο login node *(**πάντα πρέπει να επιβεβαιώνετε ότι λειτουργεί στο login node**)* με **./c1.sh**

Αν σας εμφανίσει τα 2 παραπάνω μηνύματα, τότε το script λειτουργεί σωστά στο login node και κατ' επέκταση θα λειτουργήσει σωστά στα worker nodes, αφού έχουν το ίδιο λειτουργικό σύστημα.

[1] //www.pbsworks.com/

Για την υποβολή θα πρέπει να προσδιορίσετε τους πόρους που αιτήστε. Η εργασία σας θα βρίσκεται σε κατάσταση αναμονής μέχρι να βρεθούν όλοι οι πόροι. Αν ζητήσετε υπερβολικά πολλούς πόρους (π.χ. 1000 επεξεργαστές) δε θα εκτελεστεί ποτέ, και θα βρίσκεται συνεχώς σε κατάσταση αναμονής. Ο προσδιορισμός των πόρων γίνεται είτε με παράμετρο στο **qsub** είτε με κατάλληλη αναγραφή στο αρχείο υποβολής, όπως θα δούμε στη συνέχεια.

Για να υποβάλετε την εργασία με τον πρώτο τρόπο, θα χρησιμοποιήσετε το qsub με την παράμετρο **-l** (παράμετρο ελ (**limits**)) στην οποία επιλέγετε τους πόρους που θα αιτηθείτε.

Για να αιτηθείτε μια ομάδα με 1 επεξεργαστή, θα δώσετε
qsub -l select=1:ncpus=1 ./c1.sh
και θα σας εμφανιστεί ο αριθμός καταχώρησης στο χρονοδρομολογητή,

π.χ. **124744.pbs** ή **124744.acaad01** .

Αναλόγως του φόρτου εργασίας των batch nodes, υπάρχει περίπτωση να μην ξεκινήσει αμέσως η εργασίας σας.

Μόλις ολοκληρωθεί η εργασία σας θα βρείτε στον κατάλογο εργασίας σας, δυο καινούργιο αρχεία που έχουν το ίδιο όνομα, (π.χ. c1.sh), και με κατάληξη τον αριθμό καταχώρησης, δηλαδή στο παράδειγμα μας 124744. Το πρώτο αρχείο έχει πριν τον αριθμό το γράμμα **e** (=errror) και το δεύτερο το γράμμα **o** (=output), στα οποία καταχωρείται η τυπική έξοδος λάθους (standard error) και η τυπική κανονική έξοδος (standard output) αντίστοιχα. Επειδή τα προγράμματα εκτελούνται απομακρυσμένα και δεν έχετε πρόσβαση σε εκείνους τους κόμβους, η μόνη πληροφορία της εκτέλεσης έρχεται διαμέσου αυτών των αρχείων.

Μπορείτε να δείτε τη διαθεσιμότητα των batch nodes με:

qnodes

και της ουράς εκτέλεσης με

qstat -q

Αν αναγράφονται 0 ελεύθεροι κόμβοι, τότε θα πρέπει να περιμένετε. Το **qfree** είναι ένα αρχείο φλοιού που χρησιμοποιεί το εργαλείο **qmgr** . Οι περίεργοι φοιτητές, μπορούν να δουν τις εντολές που περιέχει το script με την εντολή:

more `which qfree`

Για να διαπιστώσετε τη σειρά που έχετε, δώστε **qstat** όπου εκτυπώνεται η σειρά προτεραιότητας *(δοκιμάστε και την παράμετρο* **-a***)*. Το αναγνωριστικό workq σημαίνει work queue, δηλαδή ότι βρίσκεται σε μια ουρά εργασίας. Με τη **qdel** απομακρύνουμε την εργασία μας από το PBS.

Αν και το **qsub** δέχεται πάρα πολλές παραμέτρους, η καθιερωμένη διαδικασία είναι να δημιουργείται ένα αρχείο υποβολής PBS, στο οποίο τοποθετούνται με # οι παράμετροι εκτέλεσης στο cluster. Για παράδειγμα, η παράμετρος **#PBS -j oe**

δηλώνει ότι και η τυπική έξοδος (standard output) και η έξοδος λάθους (standard error), θα συνδυαστούν (**join**) σε ένα κοινό αρχείο στο τέλος.

Επίσης, σε περίπτωση που θέλουμε να υποβάλουμε ένα πρόγραμμα δυαδικής μορφής, θα πρέπει να δημιουργήσουμε το αρχείο υποβολής PBS. Διαφορετικά, αν προσπαθήσουμε να υποβάλουμε ένα δυαδικό (μεταγλωττισμένο) αρχείο, όπως το `./myprogram` θα εμφανιστεί: `qsub: must be an ascii script`.

Ένα παράδειγμα ενός αρχείου υποβολής για ένα εκτελέσιμο myprogram που δέχεται μια παράμετρο γραμμής εντολής και τη θέσαμε σε τιμή 64, είναι το εξής:

```
#!/bin/sh
#PBS -N myjob
#PBS -j oe
#PBS -l walltime=0:15:00
export OMP_NUM_THREADS=40
cd ~/threading
./myprogram 64
```

Το αρχείο ξεκινάει με τη δήλωση του φλοιού, ακολουθούν μια ή περισσότερες γραμμές ρύθμισης του PBS (`#PBS`), στη συνέχεια θέτουμε τις μεταβλητές περιβάλλοντος για το πρόγραμμά μας *(αν χρειάζεται),* αλλαγή καταλόγου *(αν χρειάζεται)*, και εκτελείται η εφαρμογή μας. Στο προηγούμενο παράδειγμα το `-N` δηλώνει το όνομα (**name**) της εργασίας, το `-j` δηλώνει αν και πως θα γίνει η σύνδεση των stderror και *stdoutput (το* `oe` *από τη σελίδα βοήθειας του qsub έχει ως συνέπεια "Standard error and standard output are merged into standard output")*, και το l είναι η αίτηση για συγκεκριμένο πόρο.

Δοκιμάστε να υποβάλετε το **c1.sh** με το qsub χωρίς κάποια παράμετρο. Πέτυχε ή απέτυχε η υποβολή; **(C06-7)**

Μπορείτε να τοποθετήσετε όλες τις παραμέτρους που θα δίνατε στο qsub στο ίδιο το submission αρχείο. Για παράδειγμα, δημιουργήστε το `c2.sh` με τον παρακάτω κώδικα και υποβάλετε το με το qsub, ως `qsub c2.sh`.

```
#!/bin/sh
#PBS -l select=1:ncpus=1
#PBS -N dasygenis#1
#PBS -j oe
#PBS -l walltime=0:10:00
echo "Working directory is $PWD, at `hostname`"
echo "I see disks: `mount`"
```

Θα διαπιστώσετε, ότι θα γίνει σωστά η υποβολή, αφού έχουμε ορίσει τις PBS παραμέτρους.

Όταν εκτελεστεί η εργασίας μας, θα διαπιστώσουμε ότι στον κατάλογο εργασίας έχουν δημιουργηθεί 1 ή περισσότερα αρχεία. Το κάθε αρχείο θα έχει στο όνομά του:

- το όνομα της εργασίας που είχε υποβληθεί
- το αναγνωριστικό εργασίας που είχε δημιουργηθεί, και

- το γράμμα ο *(standard output)*, ή e *(standard error)* ή και τα δυο, στα οποία τοποθετείται η τυπική κανονική έξοδος ή τυπική έξοδος σφαλμάτων.

Σε περίπτωση που θέλουμε να τροποποιήσουμε το αρχείο εξόδου standard output ή standard error θα χρησιμοποιήσουμε τις παραμέτρους -o και -e αντίστοιχα, είτε στο qsub είτε μέσα στο αρχείο υποβολής ως

#PBS -o XXXXXX και #PBS -e YYYYY.

ΠΡΟΣΟΧΗ: Σε περίπτωση που σας αναφερθεί το παρακάτω μήνυμα σφάλματος κατά την υποβολή εργασίας με το qsub, τότε σημαίνει ότι έχετε κάποιο τυπογραφικό λάθος στις γραμμές **#PBS** ή ότι δεν έχετε ορίσει σωστά το walltime.

Error allocating memory for add_verify_resources

(C06-8) Να δημιουργήσετε και να υποβάλετε στο PBS ένα πρόγραμμα σε C με όνομα c1.c το οποίο θα τυπώνει:

- "hello world" ,
- το όνομα υπολογιστή, όπως εμφανίζεται με τη συνάρτηση gethostname(), (μπορείτε να δείτε τη σύνταξη με **man gethostname**)
- Και τον αριθμό των επεξεργαστικών πυρήνων, όπως εμφανίζεται από: sysconf(_SC_NPROCESSORS_CONF) (**man sysconf**)

Σε περίπτωση που θέλουμε να χρησιμοποιήσουμε μια εργασία στο OpenMPI, τότε η υποβολή γίνεται με το αρχείο PBS όπως είδαμε παραπάνω, και δε χρειάζεται να δηλωθεί κανένα hostfile γιατί το περιβάλλον που έχει στήσει ο διδάσκων, αυτοματοποιεί αυτή τη διαδικασία.

Ένα παράδειγμα χρήσης του cluster με το OpenMPI.

Αρχικά δημιουργούμε το hellompi.c στο login node, όπως παρακάτω:

```
#include <stdio.h>
#include <mpi.h>
int main(int argc, char *argv[])
{
// Initialize the MPI environment
MPI_Init(&argc, &argv);
// Get the number of processes
int world_size;
MPI_Comm_size(MPI_COMM_WORLD, &world_size);
```

```
// Get the rank of the process
int world_rank;
MPI_Comm_rank(MPI_COMM_WORLD, &world_rank);
// Get the name of the processor
char processor_name[MPI_MAX_PROCESSOR_NAME];
int name_len;
MPI_Get_processor_name(processor_name, &name_len);
// Print off a hello world message
printf("Hello world from processor %s, rank %d"
       " out of %d processors\n",
       processor_name, world_rank, world_size);
// Finalize the MPI environment.
MPI_Finalize();
return 0;
}
```

Στη συνέχεια κάνουμε compile με το **mpicc**:

```
mpicc hellompi.c -o  hellompi
```

Αν εκτελέσουμε το **./hellompi** θα διαπιστώσουμε ότι εκτελείται κανονικά στο login node και άρα θα εκτελεστεί κανονικά και στο cluster.

```
mdasygenis@pleiades:~/MTL$ ./hellompi
Hello world from processor pleiades.icte.uowm.gr, rank 0 out of 1 processors
mdasygenis@pleiades:~/MTL$
```

Μπορούμε να εκτελέσουμε το **./hellompi** με το mpirun ώστε να χρησιμοποιηθεί και το MPI με την παράμετρο δήλωσης κόμβων –n, για να διαπιστώσουμε ότι λειτουργεί σωστά με κάποιον αριθμό κόμβων, π.χ. **mpirun –n 4 ./hellompi**

```
mdasygenis@pleiades:~/MTL$ mpirun -n 4 ./hellompi
Hello world from processor pleiades.icte.uowm.gr, rank 3 out of 4 processors
Hello world from processor pleiades.icte.uowm.gr, rank 2 out of 4 processors
Hello world from processor pleiades.icte.uowm.gr, rank 1 out of 4 processors
Hello world from processor pleiades.icte.uowm.gr, rank 0 out of 4 processors
mdasygenis@pleiades:~/MTL$
```

Από τη στιγμή που έχουμε επιβεβαιώσει την ορθή λειτουργία στο login node, είμαστε έτοιμοι να υποβάλλουμε την εργασία στο cluster. Δημιουργούμε το παρακάτω αρχείο υποβολής sub.sh:

```
#!/bin/sh
#PBS -N dasygenisjob
```

```
#PBS -j oe
#PBS -m n
#PBS -l nodes=2:ppn=4
#PBS -l walltime=00:00:30
cd $HOME/MTL
mpirun ./hellompi
```

Προσέξτε ότι πρέπει πριν την εκτέλεση **να δώσουμε cd στον κατάλογο που έχουμε το εκτελέσιμο**. Επίσης, ζητάμε 2 ξεχωριστούς κόμβους, και ο κάθε κόμβος να μας δώσει 4 διεργασίες (processes per node) που σημαίνει ότι θα πρέπει να υποστηρίζει τόσους πυρήνες (συνολικά αιτούμαστε 8 διεργασίες). Τέλος, ορίζουμε 30 δευτερόλεπτα μέγιστο χρόνο εκτέλεσης στο walltime, και με την παράμετρο –N ορίζουμε το όνομα της εργασίας σε dasygenisjob (οπότε και η έξοδος θα τοποθετηθεί σε ένα αρχείο που θα φέρει αυτό το όνομα). Να τροποποιήσετε το όνομα της εργασίας από dasygenisjob σε ένα δικό σας σε οποιοδήποτε αρχείο υποβολής από εδώ και πέρα.

Κάνουμε εκτελέσιμο το παραπάνω αρχείο (**chmod u+x sub.sh**) και το υποβάλουμε (**qsub sub.sh**).

Μια τυπική εκτέλεση είναι η παρακάτω.

```
Hello world from processor pleiades.icte.uowm.gr, rank 0 out of 8 processors
Hello world from processor pleiades.icte.uowm.gr, rank 3 out of 8 processors
Hello world from processor pleiades.icte.uowm.gr, rank 1 out of 8 processors
Hello world from processor pleiades.icte.uowm.gr, rank 2 out of 8 processors
Hello world from processor mach-three.icte.uowm.gr, rank 4 out of 8 processors
Hello world from processor mach-three.icte.uowm.gr, rank 5 out of 8 processors
Hello world from processor mach-three.icte.uowm.gr, rank 7 out of 8 processors
Hello world from processor mach-three.icte.uowm.gr, rank 6 out of 8 processors
```

Θα πρέπει να διαπιστώσετε ότι εκτελείται σε 2 διαφορετικά μηχανήματα και το κάθε μηχάνημα δίνει 4 διεργασίες (δηλαδή, έχει 4 πυρήνες).

Αν χρειαζόμαστε GPUS (για cuda), τότε πρέπει να το δηλώσουμε ως εξής:

```
#!/bin/sh
#PBS -N dasygenisjob
#PBS -j oe
#PBS -m n
#PBS -l nodes=2:ppn=2:gpus=1
#PBS -l walltime=00:00:30
cd $HOME/MTL
mpirun ./hellompi
```

Δηλαδή, θέλουμε 2 κόμβους (nodes=2), που ο κάθε κόμβος να έχει 2 πυρήνες (ppn=2) και 1 gpu (gpus=1).

Προσέξτε ότι αν ζητήσουμε παραπάνω πόρους που δεν υπάρχουν στο cluster του ΤΜΤΠ, π.χ.

```
#PBS -l nodes=2:ppn=4:gpus=1
```

τότε η εργασία θα είναι συνέχεια σε standby (Q) και ποτέ δε θα εκτελεστεί:

```
825.pleiades              dasygenisjob     mdasygenis              0 Q linux-spool
```

και μπορούμε να πληροφορηθούμε γιατί παραμένει στο Q με την εντολή **`qstat -f ID.`**

Στο πεδίο comment θα μας εμφανίσει το λόγο που δεν εκτελείται. Για παράδειγμα, στην παραπάνω εργασία, θα εμφανιστεί το μήνυμα

```
comment = Not Running: Not enough of the right type of nodes are available
```

δηλαδή δεν υπάρχουν αρκετοί πόροι για την εργασία, και θα πρέπει να τη διαγράψουμε με **qdel <ID>** (π.χ. **qdel 825**) και να την ξανα-υποβάλλουμε με τους σωστούς πόρους.

```
mdasygenis@pleiades:~/MTL$ qstat -f 825
Job Id: 825.pleiades.icte.uowm.gr
    Job_Name = dasygenisjob
    Job_Owner = mdasygenis@pleiades.icte.uowm.gr
    job_state = Q
    queue = linux-spool
    server = pleiades.icte.uowm.gr
    Checkpoint = u
    ctime = Thu Mar  3 14:35:32 2016
    Error_Path = pleiades.icte.uowm.gr:/zstorage/home/mdasygenis/MTL/dasygenis
        job.e825
    Hold_Types = n
    Join_Path = oe
    Keep_Files = n
    Mail_Points = n
    mtime = Thu Mar  3 14:35:32 2016
    Output_Path = pleiades.icte.uowm.gr:/zstorage/home/mdasygenis/MTL/dasygeni
        sjob.o825
    Priority = 0
    qtime = Thu Mar  3 14:35:32 2016
    Rerunable = True
    Resource_List.nodect = 2
    Resource_List.nodes = 2:ppn=2:gpus=3
    Resource_List.walltime = 00:00:30
    Variable_List = PBS_O_QUEUE=linux-spool,
        PBS_O_HOME=/zstorage/home/mdasygenis,PBS_O_LOGNAME=mdasygenis,
        PBS_O_PATH=/sbin:/bin:/usr/sbin:/usr/bin:/usr/games:/usr/local/sbin:/
        usr/local/bin:/zstorage/home/mdasygenis/bin:/usr/local/cuda/bin,
        PBS_O_MAIL=/var/mail/mdasygenis,PBS_O_SHELL=/bin/bash,
        PBS_O_LANG=en_US.UTF-8,PBS_O_WORKDIR=/zstorage/home/mdasygenis/MTL,
        PBS_O_HOST=pleiades.icte.uowm.gr,PBS_O_SERVER=pleiades.icte.uowm.gr
    comment = Not Running: Not enough of the right type of nodes are available

    etime = Thu Mar  3 14:35:32 2016
    submit_args = c.sh
    fault_tolerant = False
    job_radix = 0
    submit_host = pleiades.icte.uowm.gr
```

Τέλος, μια άλλη προχωρημένη λειτουργία είναι ο ορισμός συγκεκριμένων κόμβων με το όνομα, το οποίο γίνεται με

```
#PBS -l nodes=andromeda.icte.uowm.gr+alcyone.icte.uowm.gr
```

Επίσης, μπορούμε να ζητήσουμε έναν αριθμό διεργασιών από τον κόμβο με τη σύνταξη: **`#PBS -l nodes=andromeda.icte.uowm.gr:ppn=2`**

Δηλαδή, από το andromeda θέλουμε 2 διεργασίες (και κατ' επέκταση 2 πυρήνες).

Μπορούν να συνδυαστούν αυτά και ως:

```
#PBS -l nodes=andromeda.icte.uowm.gr:ppn=2+alcyone.icte.uowm.gr
```

Δηλαδή 2 διεργασίες από το andromeda και 1 από το alcyone.

Τα ονόματα και τις δυνατότητες των κόμβων της συστοιχίας φαίνονται με την εντολή **`qnodes`**.

Παρόμοια με το OpenMPI μπορούμε να χρονοδρομολογήσουμε εργασίες σε OpenMP. Επειδή το OpenMP βασίζεται σε κοινόχρηστη μνήμη στο ίδιο μηχάνημα, δεν παίζει σημασία πόσα μηχανήματα θα αιτηθούμε (παράμετρος nodes) αφού θα έχουμε μόνο εκτέλεση στον πρώτο node και οι υπόλοιποι δε θα συμμετέχουν.

Μπορούμε να το δούμε με ένα παράδειγμα. Δημιουργήστε το αρχείο hello_openmp.c ως εξής:

```
#include <omp.h>
#include <stdio.h>
#include <stdlib.h>
int main (int argc, char *argv[]) {
int nthreads, tid;
  /* Fork a team of threads giving them their own copies of variables */
#pragma omp parallel private(nthreads, tid)
  {
  /* Each thread obtains its thread number */
  tid = omp_get_thread_num();

  /* Each thread executes this print */
  printf("Hello World from thread = %d\n", tid);

  /* Only the master thread does this */
  if (tid == 0)
     {
      nthreads = omp_get_num_threads();
      printf("Total number of threads = %d\n", nthreads);
     }
  }  /* All threads join master thread and disband */
}
```

Αφού κάνετε το παραπάνω compile, δημιουργήστε το αρχείο υποβολής qsub3.sh με περιεχόμενα:

```
#!/bin/sh
#PBS -N dasygenisjob
#PBS -j oe
#PBS -m n
#PBS -l nodes=andromeda.icte.uowm.gr+alcyone.icte.uowm.gr
#PBS -l walltime=00:00:30
cd $HOME/MTL
./hello_openmp
```

Αν και χρησιμοποιούμε 2 μηχανήματα, παρατηρούμε στην έξοδο ότι χρησιμοποιούνται τα threads σε ένα μόνο κόμβο (από τους δυο 24 πύρηνους):

```
Hello World from thread = 15
Hello World from thread = 11
Hello World from thread = 16
Hello World from thread = 22
Hello World from thread = 10
Hello World from thread = 9
Hello World from thread = 4
Hello World from thread = 8
Hello World from thread = 20
Hello World from thread = 3
Hello World from thread = 12
Hello World from thread = 13
Hello World from thread = 17
Hello World from thread = 23
Hello World from thread = 19
Hello World from thread = 18
Hello World from thread = 21
Hello World from thread = 0
Hello World from thread = 1
Hello World from thread = 5
Hello World from thread = 2
Total number of threads = 24
Hello World from thread = 6
Hello World from thread = 14
Hello World from thread = 7
```

Μπορούμε να περιορίσουμε τον αριθμό των threads με τη μεταβλητή OMP_NUM_THREADS, με ένα αρχείο υποβολής όπως το παρακάτω:

```
#!/bin/sh
#PBS -N dasygenisjob
#PBS -j oe
#PBS -m n
#PBS -l nodes=andromeda.icte.uowm.gr+alcyone.icte.uowm.gr
#PBS -l walltime=00:00:30
export OMP_NUM_THREADS=12
```

```
cd $HOME/MTL
./hello_openmp
```

Στην έξοδο της εκτέλεσης, θα διαπιστώσετε ότι υπάρχουν 12 νήματα.

Σε περίπτωση που θέλουμε να χρησιμοποιήσουμε πολλαπλά νήματα σε πολλαπλούς κόμβους, τότε θα πρέπει να χρησιμοποιήσουμε και OpenMP και OpenMPI.

Τοποθετήστε τα περιεχόμενα που σας δίνονται παρακάτω στο αρχείο hello_openmpi_openmp.c

```
#include <stdio.h>
#include "mpi.h"
#include <omp.h>

int main(int argc, char *argv[]) {
  int numprocs, rank, namelen;
  char processor_name[MPI_MAX_PROCESSOR_NAME];
  int iam = 0, np = 1;

  MPI_Init(&argc, &argv);
  MPI_Comm_size(MPI_COMM_WORLD, &numprocs);
  MPI_Comm_rank(MPI_COMM_WORLD, &rank);
  MPI_Get_processor_name(processor_name, &namelen);

  #pragma omp parallel default(shared) private(iam, np)
  {
    np = omp_get_num_threads();
    iam = omp_get_thread_num();
    printf("Hello from thread %d out of %d from process %d out of %d on %s\n",
           iam, np, rank, numprocs, processor_name);
  }

  MPI_Finalize();
}
```

Η εκτέλεση θα πρέπει να γίνεται με το mpirun με το παρακάτω αρχείο υποβολής:

```
#!/bin/sh
#PBS -N dasygenisjob
#PBS -j oe
#PBS -m n
#PBS -l nodes=2
#PBS -l walltime=00:00:30
cd $HOME/MTL
mpirun ./hello_openmp_openmpi
```

Αν το εκτελέσετε θα πάρετε μια έξοδο όπως η παρακάτω, όπου φαίνεται το όνομά του κόμβου εργάτη και ο αριθμός thread.

```
Hello from thread 0 out of 2 from process 1 out of 2 on mach-two.icte.uowm.gr
Hello from thread 1 out of 2 from process 1 out of 2 on mach-two.icte.uowm.gr
Hello from thread 6 out of 16 from process 0 out of 2 on pleiades.icte.uowm.gr
Hello from thread 4 out of 16 from process 0 out of 2 on pleiades.icte.uowm.gr
Hello from thread 1 out of 16 from process 0 out of 2 on pleiades.icte.uowm.gr
Hello from thread 2 out of 16 from process 0 out of 2 on pleiades.icte.uowm.gr
Hello from thread 9 out of 16 from process 0 out of 2 on pleiades.icte.uowm.gr
Hello from thread 7 out of 16 from process 0 out of 2 on pleiades.icte.uowm.gr
Hello from thread 5 out of 16 from process 0 out of 2 on pleiades.icte.uowm.gr
Hello from thread 12 out of 16 from process 0 out of 2 on pleiades.icte.uowm.gr
Hello from thread 13 out of 16 from process 0 out of 2 on pleiades.icte.uowm.gr
Hello from thread 14 out of 16 from process 0 out of 2 on pleiades.icte.uowm.gr
Hello from thread 10 out of 16 from process 0 out of 2 on pleiades.icte.uowm.gr
Hello from thread 3 out of 16 from process 0 out of 2 on pleiades.icte.uowm.gr
Hello from thread 8 out of 16 from process 0 out of 2 on pleiades.icte.uowm.gr
Hello from thread 15 out of 16 from process 0 out of 2 on pleiades.icte.uowm.gr
Hello from thread 11 out of 16 from process 0 out of 2 on pleiades.icte.uowm.gr
Hello from thread 0 out of 16 from process 0 out of 2 on pleiades.icte.uowm.gr
```

Παρατηρήστε στην παραπάνω έξοδο, ότι δημιουργήθηκε η διεργασία 1 στο μηχάνημα mach-two με 2 νήματα και η διεργασία 0 στο μηχάνημα pleiades με 16 νήματα.

Επειδή η συστοιχία του Τμήματος αποτελείται από υπολογιστές με διάφορες αρχιτεκτονικές, προτείνεται να ζητήσετε κόμβους με ίδιο αριθμό επεξεργαστικών πυρήνων, διαφορετικά μπορεί να σας δοθεί ένα σύστημα με 24 επεξεργαστές και ένα με 2, όπως παραπάνω.

Σε περίπτωση που θέλουμε να εκτελέσουμε ένα πρόγραμμα CUDA στο cluster θα πρέπει να γίνει η κατάλληλη υποβολή σε μηχάνημα που έχει GPU όπως δείχνει το παρακάτω παράδειγμα.

Τοποθετήστε τα παρακάτω στο αρχείο hello_cuda.cu

```
// This is the REAL "hello world" for CUDA!
// It takes the string "Hello ", prints it, then passes it to CUDA with an array
// of offsets. Then the offsets are added in parallel to produce the string "World!"
// By Ingemar Ragnemalm 2010
```

```
#include <stdio.h>
const int N = 16;
const int blocksize = 16;
__global__
void hello(char *a, int *b)
{
    a[threadIdx.x] += b[threadIdx.x];
}
int main()
{
    char a[N] = "Hello \0\0\0\0\0\0";
    int b[N] = {15, 10, 6, 0, -11, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0};
    char *ad;
    int *bd;
    const int csize = N*sizeof(char);
    const int isize = N*sizeof(int);
    printf("%s", a);
    cudaMalloc( (void**)&ad, csize );
    cudaMalloc( (void**)&bd, isize );
    cudaMemcpy( ad, a, csize, cudaMemcpyHostToDevice );
    cudaMemcpy( bd, b, isize, cudaMemcpyHostToDevice );
    dim3 dimBlock( blocksize, 1 );
    dim3 dimGrid( 1, 1 );
    hello<<<dimGrid, dimBlock>>>(ad, bd);
    cudaMemcpy( a, ad, csize, cudaMemcpyDeviceToHost );
    cudaFree( ad );
    cudaFree( bd );
    printf("%s\n", a);
    return EXIT_SUCCESS;
}
```

Κάντε το compile στο Pleiades ως εξής:

```
nvcc hello_cuda.cu -o hello_cuda
```

Δημιουργήστε το παρακάτω αρχείο υποβολής, που αιτείται μια GPU:

```
#!/bin/sh
#PBS -N dasygenisjob
```

```
#PBS -j oe
#PBS -m n
#PBS -l nodes=1:ppn=1:gpus=1
#PBS -l walltime=00:00:30
cd $HOME/MTL
./hello_cuda
```

Στο αρχείο που θα δημιουργηθεί με τη τυπική έξοδο της διεργασίας, θα βρείτε το γνωστό μήνυμα hello world.

Ασφαλώς, μπορούν να γίνουν όλοι οι συνδυασμοί με OpenMP, OpenMPI και CUDA.

7. Βελτιστοποίηση κώδικα με απομάκρυνση των σημείων καθυστέρησης

Χρήση του εργαλείου Intel Vtune Amplifier 2013 για τη βελτιστοποίηση κώδικα με την απομάκρυνση των σημείων καθυστέρησης *(locks & waits)*.

Σε αυτή την εργασία θα χρησιμοποιήσουμε το εργαλείο Intel Vtune Amplifier 2013 της Intel, προκειμένου να αναλύσουμε ένα κώδικα, να βρούμε τα σημεία κώδικα τα οποία προκαλούν μεγάλη καθυστέρηση λόγω υπερβολικών κλειδωμάτων και καθυστερήσεων, να βελτιστοποιήσουμε τον κώδικα σε αυτά τα σημεία, και να διαπιστώσουμε την επίτευξη των στόχων μας.

Θα χρησιμοποιήσουμε τις αναλυτικές οδηγίες που αναγράφονται στο έγγραφο **"`locks_amplxe_lin.pdf`"**. Θα πρέπει να προσέξετε στη σύνταξη των εντολών, γιατί αν κάνετε κάποιο λάθος και αργήσετε να το παρατηρήσετε, θα πρέπει να γυρίσετε κάποιες σελίδες πίσω για να επαναλάβετε την άσκηση με τις σωστές εντολές.

Με το πέρας της άσκησης θα πρέπει να μπορείτε να απαντήσετε στις εξής ερωτήσεις:

- **(C07-1) Ποια είναι τα πρώτα βήματα για να αναλύσουμε την εφαρμογή μας, πριν εκτελέσουμε το amplifier και αμέσως μόλις το εκτελέσουμε;**
- **(C07-2) Πως αναγνωρίζουμε τα σημεία του κώδικα που προκαλούν μεγάλη καθυστέρηση λόγω υπερβολικών κλειδωμάτων συγχρονισμού;**
- **(C07-3) Πως μπορούμε άμεσα να τροποποιήσουμε τον κώδικα μέσα από το πρόγραμμα amplifier;**

7.1 Διαδικασία Εκτέλεσης Άσκησης

- Συνδεθείτε μέσω της γραφικής διεπαφής VNC στο MTL, όπως έχετε κάνει σε προηγούμενο εργαστήριο.
- Για να χρησιμοποιήσετε το εργαλείο Intel Vtune Amplifier, θα πρέπει να αρχικοποιήσετε το περιβάλλον *(δηλαδή, να τεθούν κάποιες μεταβλητές, όπως και να τροποποιηθεί η διαδρομή αναζήτησης των εκτελέσιμων εφαρμογών για να προστεθεί η τοποθεσία του amplifier)*. Το πρόγραμμα έχει εγκατασταθεί στον κατάλογο /opt/intel/vtune_amplifier_xe_2013/ . Μέσα στον κατάλογο βρίσκεται το αρχείο αρχικοποίησης amplxe-vars.sh . Δώστε:
  `source /opt/intel/vtune_amplifier_xe_2013/amplxe-vars.sh`
  (Μπορείτε να δείτε τα περιεχόμενα του με `more` *)*
- Στο συγκεκριμένο tutorial θα χρησιμοποιήσουμε τον κώδικα tachyon που βρίσκεται συμπιεσμένος στο αρχείο: `tachyon_vtune_amp_xe.tgz` μέσα στη διαδρομή:
  `/opt/intel/vtune_amplifier_xe_2013/samples/en/C++/`
- Απομακρύνετε τον κατάλογο tachyon αν τον έχετε ήδη, και αποσυμπιέστε το tachyon.

```
cd ; rm -rf tachyon ;  tar -xvzf
/opt/intel/vtune_amplifier_xe_2013/samples/en/C++/tachyon
_vtune_amp_xe.tgz
```

- Εισέλθετε στον κατάλογο tachyon που έχει δημιουργηθεί, επιβεβαιώστε ότι στο Makefile χρησιμοποιείτε το compiler gcc και όχι icc και δώστε την εντολή καθαρισμού και συμβολομετάφρασης του project

  `cd ;cd tachyon ; make clean ; make`

  Σε περίπτωση που εμφανιστεί πρόβλημα στο compilation, θα πρέπει να κάνετε τα εξής βήματα:
  (α) επεξεργάζεστε το αρχείο ~/tachyon/Makefile
  (b) βρίσκετε τη γραμμή που ξεκινάει με LIBS
  (c) προσθέτετε στο τέλος την παράμετρο `-lpthread`
- Εκτελέστε την εφαρμογή **./tachyon_analyze_locks** `dat/balls.dat` από τον κατάλογο tachyon.
- **(C07-4) Ποιος είναι ο χρόνος εκτέλεσης, όπως αναφέρεται από την εφαρμογή;**
- Συνεχίστε το tutorial, σημειώνοντας τις απαντήσεις στις ερωτήσεις με επισήμανση.
- **(C07-5) Κατά την εκτέλεση του μη βελτιστοποιημένου κώδικα, πόσος είναι ο συνολικός χρόνος CPU;**

***** Όλες οι παραπάνω απαντήσεις βρίσκονται έως και τη σελίδα 13 *****

- **Προσοχή:** Κατά τη δημιουργία του νέου project να τοποθετήσετε το σωστό working directory, όπως και τη σωστή παράμετρο, διαφορετικά δε θα μπορεί να εκτελεστεί η εφαρμογή *(και θα εμφανιστεί το μήνυμα no data ή κάτι παρόμοιο κατά την εκτέλεση της εφαρμογής).*
- **(C07-6) Κατά την εκτέλεση του μη βελτιστοποιημένου κώδικα στο εργαλείο amplifier, πόσος είναι ο συνολικός χρόνος CPU; Είναι μεγαλύτερος μικρότερος ή ίδιος με τον προηγούμενο συνολικό χρόνο εκτέλεσης;**
- **(C07-7) Η μη βελτιστοποιημένη έκδοση πόσα νήματα χρησιμοποιεί; Πως το βρήκατε;**
- **(C07-8) Στη σύνοψη των αποτελεσμάτων, σας αναφέρονται κάποιοι χρόνοι επιβάρυνσης. Ποιος είναι ο μεγαλύτερος χρόνος επιβάρυνσης;**
- **(C07-9) Ποιες είναι οι 2 πιο σημαντικές δομές συγχρονισμού, με πόσο συνολικό χρόνο καθυστέρησης, και αριθμό κλήσεων;**
- Προς το τέλος του summary υπάρχει το CPU Usage Histogram. Το πρόγραμμα έχει τοποθετήσει σε μια τιμή του οριζόντιου άξονα *(simultaneously utilized logical cpus),* μια κατακόρυφη γραμμή με όνομα "Target Concurrency". **(C07-10) Ποια είναι αυτή η τιμή και πως έχει υπολογιστεί αυτόματα για το σύστημά μας;**
- **(C07-11)** Από το ίδιο γράφημα, μπορείτε να συμπεράνετε ως προς το επίπεδο της παραλληλίας *(poor/ok/ideal/over);*
- Στο παράθυρο bottom-up:
  - **(C07-12) ποια δομή συγχρονισμού έχει το μεγαλύτερο wait time;**
  - **(C07-13) ποια δομή συγχρονισμού έχει το μεγαλύτερο wait count και πόσο επιβάρυνση σε χρόνο έχει προκαλέσει;**

***** Όλες οι παραπάνω απαντήσεις βρίσκονται έως και τη σελίδα 18 *****

Από το Bottom-up θα πατήσετε το κουμπί ανάπτυξης του mutex, δεδομένου ότι προκαλεί σημαντική επιβάρυνση, σύμφωνα με το tutorial.

Αφού πατήσετε διπλό κλικ στη συνάρτηση draw_task, θα εμφανιστεί ο κώδικας. Πατήστε το εικονίδιο που θα σας μεταφέρει στη γραμμή της συνάρτησης με τον περισσότερο CPU Time *(Go to Biggest Function Hotspot).*

- Ενεργοποιήστε την εμφάνιση της assembly, πατώντας το αντίστοιχο κουμπί. **(C07-14) Η γραμμή που έχετε επιλέξει *(με τη μεγαλύτερη επιβάρυνση)* από πόσες εντολές assembly αποτελείται; Από τις εντολές assembly, ποια είναι η εντολή με τον περισσότερο χρόνο εκτέλεσης;**

  ***** ΠΡΟΣΟΧΗ *****, αν χρησιμοποιείτε έναν κώδικα στον οποίο απουσιάζει η γραμμή 165 σύμφωνα με το tutorial, η οποία είναι pthread_mutex_lock(), τότε ήδη είναι βελτιστοποιημένος ο κώδικάς σας. Ομοίως για τη γραμμή 172.

- **(C07-15) Η συνάρτηση pthread_mutex_lock() πως λειτουργεί και γιατί έχει τοποθετηθεί στον κώδικα;**
- Στη σελίδα 21 γίνεται μια αλλαγή στον κώδικα με το να τοποθετήσετε σε σχόλια μια συνάρτηση, και να απομακρύνετε τα σχόλια από μια άλλη.
  (C07-16) Ποιο σκοπό επιτελούν αυτές οι δυο συναρτήσεις και γιατί η πρώτη συνάρτηση απαιτεί περισσότερους κύκλους CPU από τη δεύτερη;
- **(C07-17) Στο tutorial αναφέρεται ότι δε χρειάζεται το κλείδωμα γιατί η συνάρτηση είναι thread_safe. Τι σημαίνει αυτό; Χρησιμοποιήστε το Internet ή άλλη βιβλιογραφία.**

***** Όλες οι παραπάνω απαντήσεις βρίσκονται έως και τη σελίδα 22 *****

- **(C07-18) Μετά την τροποποίηση του κώδικα, και την επαναδημιουργία του εκτελέσιμου αρχείου, πόσος χρόνος CPU απαιτείται; Έχει βελτιωθεί ο χρόνος;**
- Από το “Analysis Type” πατήστε Re-resolve για να εκτελεστεί ο νέος κώδικας, και στη συνέχεια το εικονίδιο της σύγκρισης των αποτελεσμάτων.
- **(C07-19) Στο summary της σύγκρισης των αποτελεσμάτων, πόσο αναφέρεται ότι έχει μειωθεί το CPU Time;**
- **(C07-20) Στο summary η συνάρτηση που τροποποιήσατε πόσο έχει βελτιωθεί;**
- **(C07-21) Στο bottom-up της σύγκρισης *(grouping: function /call stack)*, να βρείτε τις 3 συναρτήσεις που έχουν τη μεγαλύτερη διαφορά *(βελτίωση χρόνου)* στο CPU Time.**

***** Όλες οι παραπάνω απαντήσεις βρίσκονται έως και τη σελίδα 26 *****

7.2 Χρήση διαφορετικού compiler και σύγκριση

Στο παράθυρο τερματικού, μπείτε στο φάκελο της εφαρμογής tachyon και ανοίξτε για επεξεργασία το αρχείο Makefile. Σε μια γραμμή υπάρχει η δήλωση CXX = g++ , που ορίζει το compiler. Τροποποιήστε αυτή τη γραμμή και αντί για g++ δώστε icc *(δηλαδή intel compiler)* . Αποθηκεύστε το Makefile.

Αρχικοποιήστε το περιβάλλον του Intel Compiler με το να δώσετε:

```
source /opt/intel/Compiler/latest/bin/iccvars.sh intel64
```

Επιβεβαιώστε ότι το icc βρίσκεται στη διαδρομή αναζήτησης με το να δώσετε

```
icc -v
```

και να δείτε ότι σας εμφανίζει σχετικό μήνυμα.

Απομακρύνετε τα παλαιά εκτελέσιμα αρχεία και επαναλάβετε τη διαδικασία compilation (`make clean ; make`).

Εκτελέστε την εφαρμογή με την κατάλληλη είσοδο, όπως στην αρχή της άσκησης και συγκρίνετε το χρόνο ως προς το compilation με το g++. **(C07-22) Υπάρχει διαφορά; Ποιος compiler δημιούργησε πιο βελτιωμένο εκτελέσιμο;**

Πηγαίνετε στο **Intel Vtune Amplifier → Analysis Type** και πατήστε Re-resolve. Στη συνέχεια πατήστε το κουμπί της σύγκρισης των αποτελεσμάτων (summary).
(C07-23) Τέλος, παρατηρήστε στο Bottom-up σε ποια συνάρτηση υπάρχει η καλύτερη βελτίωση.

8. Χρήση του συμβολομεταφραστή Intel® C++

Compiler XE 13.0 για τη βελτιστοποίηση εφαρμογής μέσω διανυσματοποίησης

Σε αυτή την εργασία θα χρησιμοποιήσουμε το εργαλείο Intel® C++ Compiler, προκειμένου να αναλύσουμε ένα κώδικα, να βρούμε τα σημεία κώδικα τα οποία θα μπορούσαν να βελτιστοποιηθούν μέσω διανυσματοποίησης και την εφαρμογή της τεχνικής αυτής υποβοηθούμενη από το συμβολομεταφραστή.

Θα χρησιμοποιήσουμε τις αναλυτικές οδηγίες που αναγράφονται στο έγγραφο "composerxe_linux_cmp_vec_c.pdf". Θα πρέπει να προσέξετε στη σύνταξη των εντολών, γιατί αν κάνετε κάποιο λάθος και αργήσετε να το παρατηρήσετε, θα πρέπει να γυρίσετε κάποιες σελίδες πίσω για να επαναλάβετε την άσκηση με τις σωστές εντολές.

Με το πέρας της άσκησης θα πρέπει να μπορείτε να απαντήσετε στις εξής ερωτήσεις:

- **(C08-01) Πως δημιουργούμε το baseline σενάριο, το οποίο θα χρησιμοποιηθεί για τη μέτρηση της βελτίωσης;**
- **(C08-02) Πως δημιουργείται μια αναφορά διανυσματοποίησης από το compiler;**
- **(C08-03) Πως μπορούμε να βοηθήσουμε το compiler στη διανυσματοποίηση με το να στοιχίσουμε δεδομένα, να αντικαταστήσουμε δείκτες με άλλες δομές, και να βελτιστοποιήσουμε τους βρόχους;**

8.1 Διαδικασία Εκτέλεσης Άσκησης

Μελετήστε όλο το έγγραφο ώστε να καταλάβετε από ποια βήματα θα περάσετε προκειμένου να επιτευχθεί η βελτιστοποιηση. Δώστε προσοχή στην εισαγωγή *(introduction to auto vectoriaztion)*, που εξηγεί πως λειτουργεί η διανυσματοποίηση.

- **(C08-04) Γιατί η τεχνική της διανυσματοποίησης είναι πιο αποδοτική από την απλή εκτέλεση των πράξεων;**
- **(C08-05) Μπορεί η τεχνική της διανυσματοποίησης να γίνει αυτόματα;**
- **(C08-06) Ποια παράμετρος πρέπει να χρησιμοποιηθεί στον Intel Compiler για να ενεργοποιηθεί η διανυσματοποίηση;**
- **(C08-07) Η διανυσματοποίηση μπορεί να εφαρμοστεί και σε επεξεργαστές Intel και σε άλλους επεξεργαστές;**

***** Όλες οι παραπάνω απαντήσεις βρίσκονται στην εισαγωγή *****

- Συνδεθείτε στο MTL με όποιο τρόπο θέλετε. Σε αυτή την εργασία η απλή σύνδεση τερματικού με το SSH, είναι άκρως αποτελεσματική.
- Στο συγκεκριμένο tutorial θα χρησιμοποιήσουμε τον κώδικα που βρίσκεται στη διαδρομή **<install-dir>/Samples/en_US/C++/vec_samples** ,

όπου η διαδρομή εγκατάστασης **<install-dir>** για το Manycore είναι: **/opt/intel/composer_xe_2013**

- Δημιουργήστε ένα κατάλογο στην περιοχή σας με όνομα vec_samples και αντιγράψτε όλα τα παραπάνω αρχεία.
- Για να χρησιμοποιήσετε το εργαλείο, θα πρέπει να αρχικοποιήσετε το περιβάλλον *(δηλαδή, να τεθούν κάποιες μεταβλητές, όπως και να τροποποιηθεί η διαδρομή αναζήτησης των εκτελέσιμων εφαρμογών για να προστεθεί η τοποθεσία του compiler).* Δώστε:

```
source /opt/intel/composer_xe_2013/bin/compilervars.sh intel64
```

(Μπορείτε να δείτε τα περιεχόμενα του με `more`)

ΠΡΟΣΟΧΗ: Στην εργασία αυτή ο κώδικας έχει ήδη τροποποιηθεί. Αναλόγως τις τιμές macro που δίνουμε στο compiler με την παράμετρο -D ενεργοποιείται διαφορετική ροή εκτέλεσης μέσα στο πηγαίο αρχείο. Για παράδειγμα, μέσα στο Driver.C υπάρχει μια γραμμή που αναφέρει:

```
#ifdef ALIGNED YYYY #else XXXX #endif .
```

Αυτό σημαίνει ότι αν στη γραμμή εντολής του compiler έχουμε ενεργοποιήσει το macro ALIGNED, δηλαδή αν έχουμε δώσει ….-D ALIGNED τότε θα χρησιμοποιηθεί ο κώδικας **YYYY** , διαφορετικά *(αν απουσιάζει η παραπάνω παράμετρος)* θα χρησιμοποιηθεί ο κώδικας **XXXX**.

8.2 Παράγραφος: Establishing a Performance Baseline

- Εξάγετε τις αρχικές τιμές χρόνου για το baseline σενάριο με το να εκτελέσετε την εντολή της παραγράφου Establishing a Performance Baseline.
- **(C08-08) Πόσο χρόνο αναφέρει το πρόγραμμα ότι απαιτήθηκε;**
- **(C08-09) Πόσα ήταν τα Gigaflops;**
- **(C08-10) Πως υπολογίστηκαν τα gigaflops;**
- **(C08-11) Ανοίξτε το αρχείο driver.c και απαντήστε: Ποιος κώδικας εκτελείται αν ενεργοποιηθεί το macro `NOFUNCCALL` και ποιος αν δεν ενεργοποιηθεί;**

8.3 Παράγραφος: "Generating a Vectorization Report"

(C08-12) Γιατί δε μπορούμε να χρησιμοποιήσουμε την παράμετρο παραγωγής της αναφοράς διανυσματοποίησης -vec-report1 με την παράμετρο βελτιστοποίησης O1;

- Εκτελέστε την εντολή για τη δημιουργία της αναφοράς διανυσματοποίησης.
- Εκτελέστε το νέο πρόγραμμα και σημειώστε αν βελτιώθηκε ο κώδικας:
  (C08-13) Πόσο χρόνο αναφέρει το πρόγραμμα ότι απαιτήθηκε;
  (C08-14) Πόσα ήταν τα Gigaflops;
  (C08-15) Πως υπολογίστηκαν τα gigaflops;
- **(C08-16) Πόσο επιτάχυνση στον κώδικα έχει επιτευχθεί;**
- Εκτελέστε την εντολή για τη δημιουργία αναφοράς επιπέδου 2 (περισσότερες λεπτομέρειες).
- **(C08-17) Ποια είναι τα επιπρόσθετα μηνύματα με την παράμετρο αναφοράς επιπέδου 2;**
  Από την αναφορά σημειώστε:
- **(C08-18) Ο βρόχος στη γραμμή 54 του αρχείου driver.c έχει διανυσματοποιηθεί; Δικαιολογήστε την απάντησή σας. Αν δε διανυσματοποιήθηκε να αναφέρετε το λόγο.**

8.4 Παράγραφος: "Improving Performance by Pointer Disambiguation"

- **(C08-19) Ποιο πρόβλημα δημιουργούν τα εναλλακτικά ονόματα δεικτών *(pointer alliasing);***
- **(C08-20) Πως μπορούμε να βοηθήσουμε τον compiler και να τον ενημερώσουμε ότι στον κώδικά μας δε χρησιμοποιούμε εναλλακτικά ονόματα δεικτών;**
  (C08-21) Ανοίξτε το αρχείο Multiply.c και απαντήστε: Ποιος κώδικας εκτελείται αν ενεργοποιηθεί το macro NOALIAS και ποιος αν δεν ενεργοποιηθεί;
- Εκτελέστε το νέο πρόγραμμα, αφού το κάνετε compile με το ειδικό μάκρο μη χρησιμοποίησης εναλλακτικών ονομάτων δεικτών, και σημειώστε αν βελτιώθηκε ο κώδικας.

8.5 Παράγραφος: “Improving Performance by Aligning Data”

- **(C08-22) Ποια είναι η σημασία της στοίχισης και γιατί πρέπει να μας ενδιαφέρει κατά τη διανυσματοποίηση;**
- Η στοίχιση γίνεται προς όρια 16bit, 32bit ή κάποιο άλλο και γιατί;
- **(C08-23) Ανοίξτε το αρχείο Driver.c και απαντήστε: Ποιος κώδικας εκτελείται αν ενεργοποιηθεί το macro `ALIGNED` και ποιος αν δεν ενεργοποιηθεί;**
- Εκτελέστε το νέο πρόγραμμα, αφού το κάνετε compile με το ειδικό μάκρο, και σημειώστε αν βελτιώθηκε ο κώδικας.

8.6 Παράγραφος: “Improving Performance with Interprocedural Optimization”

- **(C08-23)** Τι επιτυγχάνεται με τη δια-συναρτησιακή βελτιστοποίηση *(interprocedural optimization)*, και πως ενεργοποιείται;
- Εκτελέστε το νέο πρόγραμμα, αφού το κάνετε compile με την ειδική παράμετρο ipo, και σημειώστε αν βελτιώθηκε ο κώδικας.

8.7 Παράγραφος: “Additional Exercises”

- **(C08-24) Γιατί αν χρησιμοποιήσουμε πραγματικούς αριθμούς μονής ακρίβειας (single precission) μπορούμε να επιτύχουμε καλύτερα αποτελέσματα διανυσματοποίησης;**
  (C08-25) Ανοίξτε το αρχείο Multiply.h και απαντήστε: Ποιος κώδικας εκτελείται αν ενεργοποιηθεί το macro FTYPE=float και ποιος αν δεν ενεργοποιηθεί;
- Εκτελέστε την προηγούμενη εντολή διανυσματοποίησης, ενεργοποιώντας το makro -DFTYPE=float , ώστε να χρησιμοποιήσετε αντί για πίνακες διπλής ακρίβειας, μόνης ακρίβειας.
- Εκτελέστε το νέο πρόγραμμα και σημειώστε αν βελτιώθηκε ο κώδικας:
  (C08-26) Πόσο χρόνο αναφέρει το πρόγραμμα ότι απαιτήθηκε;
  (C08-27) Πόσα ήταν τα Gigaflops;
  (C08-28) Πως υπολογίστηκαν τα gigaflops;
- **(C08-29) Πόσο επιτάχυνση στον κώδικα έχει επιτευχθεί;**

(C08-30) Συγκεντρώστε τη διαδοχική βελτίωση που έχει επιτευχθεί:

Baseline	**Χρόνος**	**GigaFlops**	**Speedup**
Βελτιστοποίηση#1 *(pointer disambiguation)*			
Βελτιστοποίηση#2 *(data align):*			
Βελτιστοποίηση#3 *(interprocedural optimization)*			
Βελτιστοποίηση#4 *(float instead of double)*			

9. Βελτιστοποίηση κώδικα ως προς συγκεκριμένο hardware

Χρήση του εργαλείου Intel Vtune Amplifier 2013 για τη βελτιστοποίηση κώδικα ως προς συγκεκριμένο hardware *(επεξεργαστή και κρυφή μνήμη).*

Σε αυτή την εργασία θα χρησιμοποιήσουμε το εργαλείο Intel Vtune Amplifier 2013 της Intel, προκειμένου να αναλύσουμε ένα κώδικα, να βρούμε τα σημεία κώδικα τα οποία προκαλούν μεγάλη επιβάρυνση στο συγκεκριμένο hardware *(λόγω υψηλών αστοχιών στην κρυφή μνήμη)*, να βελτιστοποιήσουμε τον κώδικα σε αυτά τα σημεία, και να διαπιστώσουμε την επίτευξη των στόχων μας.

Θα χρησιμοποιήσουμε τις αναλυτικές οδηγίες που αναγράφονται στο έγγραφο "`hw_issues_amplxe_lin.pdf`". Θα πρέπει να προσέξετε στη σύνταξη των εντολών, γιατί αν κάνετε κάποιο λάθος και αργήσετε να το παρατηρήσετε, θα πρέπει να γυρίσετε κάποιες σελίδες πίσω για να επαναλάβετε την άσκηση με τις σωστές εντολές.

Με το πέρας της άσκησης θα πρέπει να μπορείτε να απαντήσετε στις εξής ερωτήσεις:

- **(C09-01) Ποια είναι τα πρώτα βήματα για να αναλύσουμε την εφαρμογή μας, πριν εκτελέσουμε το amplifier και αμέσως μόλις το εκτελέσουμε;**
- **(C09-02) Πως αναγνωρίζουμε τα σημεία του κώδικα που προκαλούν μεγάλη επιβάρυνση στο συγκεκριμένο hardware;**
- **(C09-03) Πως μπορούμε άμεσα να τροποποιήσουμε τον κώδικα μέσα από το πρόγραμμα amplifier;**

9.1 Διαδικασία Εκτέλεσης Άσκησης

- Συνδεθείτε μέσω της γραφικής διεπαφής VNC στο MTL, όπως έχετε κάνει σε προηγούμενο εργαστήριο.
- Για να χρησιμοποιήσετε το εργαλείο Intel Vtune Amplifier, θα πρέπει να αρχικοποιήσετε το περιβάλλον *(δηλαδή, να τεθούν κάποιες μεταβλητές, όπως και να τροποποιηθεί η διαδρομή αναζήτησης των εκτελέσιμων εφαρμογών για να προστεθεί η τοποθεσία του amplifier)*. Το πρόγραμμα έχει εγκατασταθεί στον κατάλογο /opt/intel/vtune_amplifier_xe_2013/ . Μέσα στον κατάλογο βρίσκεται το αρχείο αρχικοποίησης amplxe-vars.sh . Δώστε:

```
source /opt/intel/vtune_amplifier_xe_2013/amplxe-vars.sh
```

(Μπορείτε να δείτε τα περιεχόμενα του με `more`)

- Στο συγκεκριμένο tutorial θα χρησιμοποιήσουμε τον κώδικα matrix που βρίσκεται συμπιεσμένος στο αρχείο: **`matrix_vtune_amp_xe.tgz`** μέσα στη διαδρομή `/opt/intel/vtune_amplifier_xe_2013/samples/en/C++/` Αποσυμπιέστε το αρχείο αυτό στον κατάλογο εργασίας *(η εντολή εκτελείται σε μια γραμμή):*

```
cd ; tar -xvzf
/opt/intel/vtune_amplifier_xe_2013/samples/en/C++/matrix_
vtune_amp_xe.tgz
```

- Εισέλθετε στον κατάλογο matrix που έχει δημιουργηθεί και δώστε την εντολή συμβολομετάφρασης του project
  `cd matrix/linux ; make`
- Εκτελέστε μια φορά το πρόγραμμα ως ./matrix.gcc αν έχετε χρησιμοποιήσει τον compiler gcc ή ./matrix.icc αν έχετε χρησιμοποιήσει τον compiler icc. Σημειώστε:
- **(C09-04) Πόσα threads αναφέρει η εφαρμογή ότι χρησιμοποίησε;**
- **(C09-05) Ποιος είναι ο χρόνος εκτέλεσης, όπως αναφέρεται από την εφαρμογή;**
- Συνεχίστε το tutorial, σημειώνοντας τις απαντήσεις στις ερωτήσεις με επισήμανση.
- **(C09-06) Ποιος compiler χρησιμοποιήθηκε για τον κώδικα; Ο gcc ή ο icc;**
- **(C09-07) Κατά την εκτέλεση του μη βελτιστοποιημένου κώδικα, πόσος είναι ο συνολικός χρόνος CPU;**

***** Όλες οι παραπάνω απαντήσεις βρίσκονται έως και τη σελίδα 13 *****

- **Προσοχή:** Κατά τη δημιουργία του νέου project να τοποθετήσετε το σωστό working directory, όπως και τη σωστή παράμετρο, διαφορετικά δε θα μπορεί να εκτελεστεί η εφαρμογή *(και θα εμφανιστεί το μήνυμα no data ή κάτι παρόμοιο κατά την εκτέλεση της εφαρμογής).*
- **(C09-08) Το εργαλείο που χρησιμοποιείτε σε αυτό το εργαστήριο ποιες αναλύσεις υποστηρίζει ειδικά για τους επεξεργαστές της μικροαρχιτεκτονικής Nehalem/Westmere;**

  ******ERROR ***** NO PERMISSIONS *******

- **(C09-09) Κατά την εκτέλεση του μη βελτιστοποιημένου κώδικα στο εργαλείο amplifier, πόσος είναι ο συνολικός χρόνος CPU; Είναι μεγαλύτερος μικρότερος ή ίδιος με τον προηγούμενο συνολικό χρόνο εκτέλεσης;**
- **(C09-10) Η μη βελτιστοποιημένη έκδοση πόσα νήματα χρησιμοποιεί; Πως το βρήκατε;**
- **(C09-11) Ποιοι είναι οι χρόνοι CPU Time που μετρήσατε στο σύστημά σας, για τις τρεις πιο χρονοβόρες συναρτήσεις;**
- Προς το τέλος του summary υπάρχει το CPU Usage Histogram. Το πρόγραμμα έχει τοποθετήσει σε μια τιμή του οριζόντιου άξονα *(simultaneously utilized logical cpus),* μια κατακόρυφη γραμμή με όνομα "Target Concurrency". **(C09-12) Ποια είναι αυτή η τιμή και πως έχει υπολογιστεί αυτόματα για το σύστημά μας;**

- **(C09-13) Στο παράθυρο bottom-up, ποια συνάρτηση κάλεσε το initialize_2D_buffer;**
- **(C09-14) Στο παράθυρο bottom-up, ποια συνάρτηση κάλεσε το sphere_intersect;**

- **(C09-15) Στο παράθυρο bottom-up, στα δεξία υπάρχει το call stack. Ποια συνάρτηση κάλεσε την initialize_2D_buffer, σε ποια αρχείο και σε ποια γραμμή βρίσκεται;**

***** Όλες οι παραπάνω απαντήσεις βρίσκονται έως και τη σελίδα 17 *****

- Στο timeline στο παράθυρο bottom-up, αν αφήσουμε το mouse πάνω από το γράφημα, θα μας αναφερθεί ότι το utilization είναι περίπου 98% με 100%. Εντούτοις, το πρόγραμμα αναφέρει ότι το πρόγραμμα έχει “poor” cpu utilization. **(C09-16) Γιατί γίνεται αυτό; Ποια είναι η μέγιστη χρήση επί τις % για το σύστημα MTL;**
- Αφού πατήσετε διπλό κλικ στη συνάρτηση, θα εμφανιστεί ο κώδικας. Πατήστε το εικονίδιο που θα σας μεταφέρει στη γραμμή της συνάρτησης με τον περισσότερο CPU Time. *(Go to Biggest Function Hotspot).*
- Ενεργοποιήστε την εμφάνιση της assembly, πατώντας το αντίστοιχο κουμπί. **(C09-17) Η γραμμή που έχετε επιλέξει *(με τη μεγαλύτερη επιβάρυνση)* από πόσες εντολές assembly αποτελείται; Από τις εντολές assembly, ποια είναι η εντολή με τον περισσότερο χρόνο εκτέλεσης;**
- Στη σελίδα 22 γίνεται μια αλλαγή στον κώδικα με το να τοποθετήσετε σε σχόλια μια συνάρτηση, και να απομακρύνετε τα σχόλια από μια άλλη.
  (C09-18) Ποιο σκοπό επιτελούν αυτές οι δυο συναρτήσεις και γιατί η πρώτη συνάρτηση απαιτεί περισσότερους κύκλους CPU από τη δεύτερη;
  ***** Όλες οι παραπάνω απαντήσεις βρίσκονται έως και τη σελίδα 22 *****

- **(C09-19) Μετά την τροποποίηση του κώδικα, και την επαναδημιουργία του εκτελέσιμου αρχείου, πόσος χρόνος CPU απαιτείται; Έχει βελτιωθεί ο χρόνος;**
- Από το “Analysis Type” πατήστε Re-resolve για να εκτελεστεί ο νέος κώδικας, και στη συνέχεια το εικονίδιο της σύγκρισης των αποτελεσμάτων.
- **(C09-20) Στο summary της σύγκρισης των αποτελεσμάτων, πόσο αναφέρεται ότι έχει μειωθεί το CPU Time;**
- **(C09-21) Στο summary η συνάρτηση initialize_2D_buffer πόσο έχει μειωθεί σε CPU Time;**
- **(C09-22) Στο bottom-up της σύγκρισης *(grouping: function /call stack),* να βρείτε τις 3 συναρτήσεις που έχουν τη μεγαλύτερη διαφορά *(βελτίωση χρόνου)* στο CPU Time: Difference by Utilization.**
  ***** Όλες οι παραπάνω απαντήσεις βρίσκονται έως και τη σελίδα 26 *****

9.2 Χρήση διαφορετικού compiler και σύγκριση

Στο παράθυρο τερματικού, μπείτε στο φάκελο της εφαρμογής tachyon και ανοίξτε για επεξεργασία το αρχείο Makefile. Σε μια γραμμή υπάρχει η δήλωση CXX = g++ , που ορίζει το compiler. Τροποποιήστε αυτή τη γραμμή και αντί για g++ δώστε icc *(δηλαδή intel compiler)* . Αποθηκεύστε το Makefile.

Αρχικοποιήστε το περιβάλλον του Intel Compiler με το να δώσετε:

```
source /opt/intel/Compiler/latest/bin/iccvars.sh intel64
```

Επιβεβαιώστε ότι το icc βρίσκεται στη διαδρομή αναζήτησης με το να δώσετε

```
icc -v
```

και να δείτε ότι σας εμφανίζει σχετικό μήνυμα.

Απομακρύνετε τα παλαιά εκτελέσιμα αρχεία και επαναλάβετε τη διαδικασία compilation (`make clean ; make`).

Εκτελέστε την εφαρμογή με την κατάλληλη είσοδο, όπως στην αρχή της άσκησης και συγκρίνετε το χρόνο ως προς το compilation με το g++.

(C09-23) Υπάρχει διαφορά; Ποιος compiler δημιούργησε πιο βελτιωμένο εκτελέσιμο;

Πηγαίνετε στο Intel Vtune Amplifier => Analysis Type και πατήστε Re-resolve. Στη συνέχεια πατήστε το κουμπί της σύγκρισης των αποτελεσμάτων (summary).

(C09-24) Τέλος, παρατηρήστε στο Bottom-up σε ποια συνάρτηση υπάρχει η καλύτερη βελτίωση.