ΠΑΝΕΠΙΣΤΗΜΙΟ
ΔΥΤΙΚΗΣ ΜΑΚΕΔΟΝΙΑΣ

# Συστήματα Παράλληλης και Κατανεμημένης Επεξεργασίας

**Ενότητα:** ΕΡΓΑΣΤΗΡΙΑΚΗ ΑΣΚΗΣΗ No:17

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 ..... 4
2. Παραδοτέα ..... 4
3. Πρόσβαση στο Grid ..... 4
4. Ολοκλήρωση διαδικασίας ..... 4
5. Δημιουργία προϋποθέσεων υποβολής εργασίας ..... 6
   5.1 Το αρχείο mpi-hooks.sh ..... 6
   5.2 Το αρχείο mpi-start-wrapper.sh ..... 7
   5.3 Το αρχείο openmpi.jdl ..... 8
6. Υποβολή της Εργασίας ..... 9

# 1. Σκοπός της άσκησης

- Πρόσβαση στο HellasGrid, δημιουργία αίτησης, δημιουργία ομάδας, ανάπτυξη, εκτέλεση, αποσφαλμάτωση, μέτρηση επιδόσεων.

# 2. Παραδοτέα

**(A)** 11 ερωτήσεις

**(C)** 5 ασκήσεις

# 3. Πρόσβαση στο Grid

Για να αποκτήσετε πρόσβαση στις υπηρεσίες του **HellasGrid** χρειάζεται:

1. Να αποκτήσετε ένα προσωπικό ψηφιακό πιστοποιητικό από την Αρχή Πιστοποίησης HellasGrid CA
   https://access.hellasgrid.gr/register/registration_form
   Αφού καταχωρήσετε τα στοιχεία σας πρέπει να απευθυνθείτε στην Α.Τ. του πανεπιστημίου στον κ. Παναγιώτη Βουτσκίδη *(pvoutskidis at uowm.gr)* στο κτήριο της Διοικούσας, έχοντας μαζί το πάσο σας.

2. Να αποκτήσετε πρόσβαση σε κάποιο User Interface της HellasGrid υποδομής και

3. Να εγγραφείτε σε ένα ή περισσότερα Virtual Organizations (VOs) ανάλογα με το επιστημονικό πεδίο που ανήκετε.

Πιο αναλυτικά μπείτε στη σελίδα:
goc.grid.auth.gr/wiki/bin/view/Groups/ALL/GridUserRegistration#User_Interface και ακολουθήστε τις οδηγίες.

Εκτελέστε τις οδηγίες μέχρι την εγγραφή στο South Eastern European Virtual Organization.

# 4. Ολοκλήρωση διαδικασίας

Αφού έχει δημιουργηθεί το username και το password

- συνδεθείτε στο μηχάνημα μέσω putty δίνοντας στο Host name τα στοιχεία που έχετε λάβει από το mail του UI,
  **`Username@Hostname`** και δώστε τον κωδικό σας όταν ζητηθεί.
- Σειρά έχει η εξαγωγή του πιστοποιητικού.

Θα πρέπει να πάτε στην καρτέλα ρυθμίσεων του browser και να επιλέξετε Πιστοποιητικά *(Certificates)*. Στην καρτέλα **'Προσωπικά Πιστοποιητικά'** θα πρέπει να εμφανίζεται το προσωπικό σας πιστοποιητικό που είναι υπογεγραμμένο από την Α.Π. HellasGrid. Αφού το μαρκάρετε θα πρέπει να επιλέξετε Εξαγωγή *(ή Backup).*

Ονομάστε το αρχείο **mycertificate**.

Το αρχείο που θα εξαχθεί από τον browser θα πρέπει να έχει κατάληξη **.pfx** ή .**p12**.

*Για όσους έχουν επιλέξει ως browser το Mozilla Firefox η διαδικασία είναι η εξής:*

Επιλογές**...** → στη καρτέλα Για προχωρημένους → Προβολή πιστοποιητικών,

επιλέξτε το πιστοποιητικό που είναι υπογεγραμμένο από την Α.Π. HellasGrid και πατήστε Αντίγραφο ασφαλείας.

**Προσέξτε** κατά την εξαγωγή του πιστοποιητικού μην πατήσετε **'εξαγωγή'** όπως αναφέρεται εάν το .p12 δεν είναι διαθέσιμο, αλλά φτιάξετε ένα αντίγραφο ασφαλείας.

- Μεταφέρετε το **mycertificate.p12** στο χώρο εργασίας.

Δώστε `ls –a` και ελέγξτε εάν υπάρχει ο κρυφός κατάλογος .globus. Εάν δεν υπάρχει δημιουργήστε τον με mkdir .globus

**(A1)** Ξαναδώστε ls –a και επαληθεύστε.

- Εφόσον έχετε βεβαιωθεί ότι το αρχείο **mycerticifate.p12** και ο κατάλογος .globus υπάρχουν ανάμεσα στα αποτελέσματα της ls -a μπορείτε να προχωρήσετε στην εκτέλεση των παρακάτω 4 εντολών μέσω των οποίων θα φτιαχτούν 2 νέα αρχεία, το **userkey.pem** *(προσωπικό κλειδί)* και το **usercert.pem** *(ψηφιακό πιστοποιητικό).*

- `openssl pkcs12 -nocerts -in mycertificate.p12 -out .globus/userkey.pem`
- `chmod 600 .globus/userkey.pem`
- `openssl pkcs12 -clcerts -nokeys -in mycertificate.p12 -out .globus/usercert.pem`
- `chmod 644 .globus/usercert.pem`

Δώστε στη συνέχεια την παρακάτω εντολή η οποία δείχνει τις ημερομηνίες ισχύος του πιστοποιητικού:

- `openssl x509 -dates -noout -in .globus/usercert.pem`

**(A2)** Screenshot ότι όλα εκτελέστηκαν χωρίς σφάλματα.

# 5. Δημιουργία προϋποθέσεων υποβολής εργασίας

Αφού έχει ολοκληρωθεί η διαδικασία της εγγραφής πρέπει να υπάρχει το ψηφιακό πιστοποιητικό στο φάκελο **~/.globus/** του User Interface *(όπως αναφέρεται στις οδηγίες)*. Η υποβολή των εργασιών γίνεται μέσα από το User Interface, ένα σύστημα που "τρέχει" Scientific Linux και χρησιμεύει σαν ένα μέσο πρόσβασης στους υπολογιστικούς και αποθηκευτικούς πόρους της πλεγματικής υποδομής.

- Αρχικά πρέπει να γίνει σύνδεση στο UI **(ui01.isabella.grnet.gr)** μέσω ssh και να δημιουργηθεί **ένα πιστοποιητικό διαμεσολάβησης** *(proxy certificate)* το οποίο εξασφαλίζει την προσωρινή πρόσβαση στην υποδομή, άρα και την υποβολή εργασιών.

- Στη συνέχεια πρέπει να δημιουργηθεί ένα αρχείο που να περιγράφει την υπολογιστική εργασία και τις ανάγκες αυτής (όπως πχ το είδος της εργασίας). Το αρχείο αυτό, που είναι απαραίτητο για την επικοινωνία της εργασίας με το Grid, είναι γραμμένο και σε μια ειδική γλώσσα που ονομάζεται Job Description Language ή απλά JDL.

- Θεωρώντας δεδομένο ότι η εργασία έχει ρυθμιστεί σωστά και είναι έτοιμη να εκτελεστεί, υποβάλλεται στην υποδομή και μπαίνει σε ουρά προτεραιότητας. Μετά την υποβολή της, ο χρήστης μπορεί να ελέγξει της κατάσταση της εργασίας, να την ακυρώσει αν το επιθυμεί και αν έχει προνοήσει, μπορεί να παρακολουθεί και την έξοδο της εφαρμογής.

- Το τελευταίο στάδιο περιλαμβάνει την περισυλλογή των αποτελεσμάτων από τον κόμβο που εκτελέστηκε η εργασία *(Worker Node)* στο **UI**. Ανάλογα με το μέγεθος των δεδομένων μπορεί να γίνει χρήση και των Storage Elements σαν ενδιάμεσο αποθηκευτικό μέσο.

Τα παραπάνω αποτελούν έναν ενδεικτικό οδηγό και ο κάθε χρήστης μπορεί να ακολουθήσει την διαδικασία που τον εξυπηρετεί καλύτερα.

Για να εκτελέσουμε ένα πρόγραμμα σε OpenMPI, πρέπει πρώτα να θέσουμε τις μεταβλητές περιβάλλοντος στις σωστές τιμές.

## 5.1 Το αρχείο mpi-hooks.sh

Δημιουργήστε το αρχείο **mpi-hooks.sh** και αντιγράψτε τον παρακάτω κώδικα:

```
#!/bin/sh
#
# This function will be called before the MPI executable is started.
# You can, for example, compile the executable itself.
#

pre_run_hook () {
I2G_MPI_APPLICATION=helloworld-MPI
# Compile the program.
echo "Compiling ${I2G_MPI_APPLICATION}"
# Actually compile the program.
```

```
cmd="mpicc ${MPI_MPICC_OPTS} -o ${I2G_MPI_APPLICATION}
${I2G_MPI_APPLICATION}.c"
echo $cmd
$cmd
if [ ! $? -eq 0 ]; then
echo "Error compiling program. Exiting..."
exit 1
fi
# Everything's OK.
echo "Successfully compiled ${I2G_MPI_APPLICATION}"
return 0
}

#
# This function will be called before the MPI executable is finished.
# A typical case for this is to upload the results to a storage element.
#
post_run_hook () {
echo "Executing post hook."
echo "Finished the post hook."
return 0
}
```

## 5.2 Το αρχείο mpi-start-wrapper.sh

Το παραπάνω script δε πρέπει να αλλάζει από χρήστες. Γι'αυτό θα δημιουργήσουμε τώρα άλλο ένα script με όνομα **mpi-start-wrapper.sh**, στο οποίο ο χρήστης θα προσθέτει τις εντολές που θέλει να εκτελέσει πριν και μετά το MPI πρόγραμμα.

```
#!/bin/bash
# Pull in the arguments.
MY_EXECUTABLE=`pwd`/$1
MPI_FLAVOR=$2

# Convert flavor to lowercase for passing to mpi-start.
MPI_FLAVOR_LOWER=`echo $MPI_FLAVOR | tr '[:upper:]' '[:lower:]'`

# Pull out the correct paths for the requested flavor.
eval MPI_PATH=`printenv MPI_${MPI_FLAVOR}_PATH`

# Ensure the prefix is correctly set.  Don't rely on the default.
eval I2G_${MPI_FLAVOR}_PREFIX=$MPI_PATH
export I2G_${MPI_FLAVOR}_PREFIX

# Touch the executable.  It exist must for the shared file system check.
# If it does not, then mpi-start may try to distribute the executable
# when it shouldn't.
touch $MY_EXECUTABLE

# Setup for mpi-start.
export I2G_MPI_APPLICATION=$MY_EXECUTABLE
export I2G_MPI_APPLICATION_ARGS=
export I2G_MPI_TYPE=$MPI_FLAVOR_LOWER
export I2G_MPI_PRE_RUN_HOOK=mpi-hooks.sh
export I2G_MPI_POST_RUN_HOOK=mpi-hooks.sh

# If these are set then you will get more debugging information.
export I2G_MPI_START_VERBOSE=1
export I2G_MPI_START_DEBUG=1

# Invoke mpi-start.
$I2G_MPI_START
```

Ο συμβολισμός **$1** αναφέρεται στα περιεχόμενα του **Argument** που βρίσκεται στο JDL αρχείο. Σε αυτή την περίπτωση είναι το όνομα του δυαδικού αρχείου. Το παραπάνω script αρχικά καλεί το mpi-start.sh, το οποίο θέτει τις μεταβλητές περιβάλλοντος για την εργασία μας. Στη συνέχεια εκτελεί το δυαδικό αρχείο. Μετά εκτελείται ο κώδικας. Τέλος ο χρήστης μπορεί να προσθέσει εντολές που θα εκτελεστούν μετά τον κώδικα, όπως να συμπιέσει τα αποτελέσματα της εκτέλεσης και να τα αποθηκεύσει στον SE.

## 5.3 Το αρχείο openmpi.jdl

Τέλος δημιουργήστε το αρχείο **openmpi.jdl** το οποίο θα περιέχει τις απαιτούμενες πληροφορίες που χρειάζεται το middleware. Παράδειγμα ενός JDL δίνεται παρακάτω:

```
#
# mpi-test.jdl
#
JobType = "MPICH";
CPUNumber = 16;
Executable = "mpi-start-wrapper.sh";
Arguments = "Hello_mpi OPENMPI";
StdOutput = "mpi-test.out";
StdError = "mpi-test.err";
InputSandbox = {"mpi-start-wrapper.sh","mpi-hooks.sh","Hello_mpi.c"};
OutputSandbox = {"mpi-test.err","mpi-test.out"};
Requirements =
Member("MPI-START", other.GlueHostApplicationSoftwareRunTimeEnvironment)
&& Member("OPENMPI", other.GlueHostApplicationSoftwareRunTimeEnvironment)
# && RegExp("grid.*.lal.in2p3.fr.*sdj$",other.GlueCEUniqueID)
;
## - the end
#
```

Είναι απαραίτητο να καθοριστεί το **JobType** ως MPICH, και να οριστεί ο αριθμός των επεξεργαστών στους οποίους θέλουμε να εκτελέσουμε τα αρχεία μας. Να σημειωθεί ότι σε περίπτωση που καθοριστεί μεγάλος αριθμός επεξεργαστών, η εκτέλεση της εργασίας θα καθυστερήσει. Αυτό θα οφείλεται πιθανότατα, στο γεγονός ότι η εργασία δε θα εκτελείται έως ότου σας έχουν ανατεθεί όλοι οι επεξεργαστές. Το script έχει οριστεί ως εκτελέσιμο.

Η γραμμή **Arguments** καθορίζει ποιο επιχείρημα πρέπει να χρησιμοποιηθεί κατά την εκτέλεση του εκτελέσιμου και στην περίπτωσή μας αυτό πρόκειται να είναι το δυαδικό και θα ανατεθεί στο **$1** που θα βρεθεί στο script. Το **Requirements** οδηγεί την εργασία σε ένα υπολογιστικό στοιχείο που υποστηρίζει MPICH2 εργασίες. Στο **InputSandbox** περιλαμβάνουμε τα αρχεία που θα σταλούν στους Worker Nodes. Στο παράδειγμά μας αποστέλλουμε τα δύο σενάρια και το εκτελέσιμο.

**(C1)** Δημιουργήστε και εκτελέστε ένα απλό MPI πρόγραμμα, το οποίο θα εμφανίζει το rank και το communicator size. Το εκτελέσιμο αρχείο πρέπει να έχει όνομα **Hello_mpi**. Εάν δεν έχει αυτό το όνομα πρέπει να τροποποιήσετε κατάλληλα το JDL αρχείο. Εκτελέστε το μία φορά με την εντολή **./Hello_mpi** .

## 6. Υποβολή της Εργασίας

Κατά την υποβολή μιάς MPI εργασίας λαμβάνουμε τα αποτελέσματα και χειριζόμαστε την υποβολή με τις εξής εντολές.

| | |
|---|---|
| **voms-proxy-info –all** | Εμφανίζει πληροφορίες για το proxy. |
| **voms-proxy-info –voms** | Δημιουργεί ένα πιστοποιητικό proxy σε έναν VO. |
| **glite-wms-job-list-match –a** | Εμφανίζει τους διαθέσιμα Υπολογιστικά Στοιχεία. |
| **glite-wms-job-submit -o –a** | Υποβολή της εργασίας και αποθήκευση του id της στο αρχείο. |
| **glite-wms-job-status –l** | Έλεγχος της κατάστασης της εργασίας |
| **glite-wms-job-cancel –l** | Ακύρωση μιας εργασίας που έχει υποβληθεί |
| **glite-wms-job-output -i --dir ./** | Ανάκτηση των αποτελεσμάτων στον κατάλογο εργασίας |

Με τις παραπάνω εντολές επιτυγχάνεται η υποβολή μιας εργασίας στο **HellasGrid**. Εκτελέστε τις εντολές για να υποβάλετε την εργασία σας. Ψάξτε στις σελίδες βοηθείας και απαντήστε:

**(A3)** Την πλήρη εντολή δημιουργίας του πιστοποιητικού στον **SEE Virtual Organization.**

**(A4)** Την πλήρη εντολή εμφάνισης των διαθέσιμων υπολογιστικών στοιχείων.

**(A5)** Την πλήρη εντολή υποβολής της εργασίας στο grid. Μπορούμε να δούμε το id της εργασίας με κάποιο τρόπο;

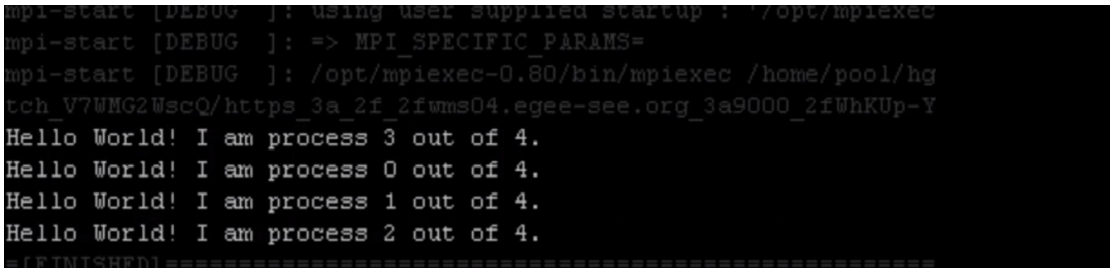
**(A6)** Την πλήρη εντολή ελέγχου της κατάστασης της εργασίας που έχουμε υποβάλει.

**(A7)** Σε περίπτωση που θέλουμε να ακυρώσουμε την εργασία που έχουμε υποβάλει, ποια εντολή χρησιμοποιούμε;

**(A8)** Την πλήρη εντολή ανάκτησης των αποτελεσμάτων της εργασίας που έχουμε υποβάλει. Ποια πρέπει να είναι η κατάσταση της εργασίας προκειμένου να μας είναι διαθέσιμα τα αποτελέσματα;

**(A9)** Επιβεβαιώστε ότι το πρόγραμμα Hello_mpi έτρεξε σωστά με screenshot που να εμφανίζονται τα σωστά αποτελέσματα. Που βρήκατε αυτά τα αποτελέσματα;

Αφού έχουν εκτελεστεί όλα σωστά στο αρχείο **.out** θα πρέπει να βρίσκονται οι εξής γραμμές.




Εάν δε θέλετε να εμφανίζονται τόσες λεπτομέρειες στο αρχείο των αποτελεσμάτων, πηγαίνετε στο αρχείο mpi-start-wrapper.sh και αφαιρέστε τις γραμμές:

export I2G_MPI_START_VERBOSE=1

export I2G_MPI_START_DEBUG=1.

**(C2)** Δημιουργήστε ένα παράλληλο πρόγραμμα εύρεσης του π (3,141593). Κάνετε τις απαραίτητες αλλαγές στο JDL αρχείο. Αναθέστε το πρόγραμμα σε 10 πόρους. Χρονομετρήστε το και βρείτε ποιός είναι ο βέλτιστος αριθμός CPU που χρειάζεται. Βρείτε και το speedup (Χρόνος σειριακού / Χρόνος παράλληλου). Η σειριακή του έκδοση δίνεται παρακάτω.

```
#include <stdio.h>
int main(){
int i, N=1024;
float pi, W=1.0/N;
pi=0.0;
for (i=0;i<N;i++)
pi+=4*W/(1+(i+0.5)*(i+0.5)*W*W);
printf("pi=%f\n",pi);
return 0;
}
```

**(A10)** Παραδώστε ένα αντίγραφο του JDL που χρησιμοποιήσατε.

Σε προγράμματα που απαιτούν μεγάλους υπολογιστικούς πόρους και ελάχιστη επικοινωνία μεταξύ των διεργασιών, η χρήση του Grid γίνεται περισσότερο αποδοτική.

**(C3)** Υλοποιήστε τη σειριακή έκδοση του προγράμματος πολλαπλασιασμού πινάκων με διαστάσεις **A[MxN]** και **B[KxL]**, γνωρίζοντας ότι πρέπει να ισχύει **N=L** και ότι το γινόμενό τους **C[MxL]** προκύπτει από τον τύπο:

$$c_{ij} = \sum_{k=1}^{L} a_{ik} b_{kj} \quad \text{με} \quad i=1,2,\ldots,L \text{ και } j=1,2,3,\ldots,M$$

Δηλαδή,

για κάθε γραμμή του πίνακα A

για κάθε στήλη του πίνακα B

κάθε στήλη του A *(δηλαδή, κάθε στοιχείο της γραμμής)*

=> υπολόγισε το `c[i][j]`.

**(C4)** Υλοποιήστε την παράλληλη έκδοση του **C3**, χρησιμοποιώντας τις παρεμποδιστικές συναρτήσεις **MPI_Send** και **MPI_Recv**. Μετρήστε το χρόνο εκτέλεσης του προγράμματος, χρησιμοποιώντας τη συνάρτηση **clock_gettime()** με τις κατάλληλες παραμέτρους.

**(C5)** Υλοποιήστε τώρα την παράλληλη έκδοση του **C3**, χρησιμοποιώντας αντί για τις παρεμποδιστικές συναρτήσεις, τις συναρτήσεις συλλογικής επικοινωνίας. Συγκεκριμένα θα χρησιμοποιείται η **MPI_Bcast** για να σταλεί ο πίνακας B και η **MPI_Scatter** για τη διασπορά του πίνακα A. Ο περιορισμός είναι ότι οι γραμμές του πίνακα A είναι ακέραιο πολλαπλάσιο του αριθμού των διεργασιών. Μετρήστε και πάλι το χρόνο εκτέλεσης.

**(A11)** Ποιο από τα δύο προγράμματα είναι ταχύτερο και γιατί πιστεύετε πως ισχύει αυτό;