Οδηγίες Χρήσης του HPC Cluster Argo για MPI και MPI+OpenMp (2020-21)

Το cluster ARGO αποτελείται α) από τον **front-end ή master node** (απλός desktop Dell OptiPlex 7050, Quad-Core Intel-Core i5-6500 @3.20GHz,16GB DDR4 2.4GHz) στον οποίο συνδεόμαστε απομακρυσμένα και β) από **11 compute nodes** σε rack για την εκτέλεση παράλληλων προγραμμάτων. Master και compute nodes συνδέονται με ethernet 1G. Οι 10 compute nodes argo-c0..9 είναι ίδιοι Dell PowerEdge 1950 (με 8 πυρήνες - 2 επεξεργαστές, ο καθένας με 4 πυρήνες XEON E5340 @2.66GHz , RAM 16GB DDR2 667MHz, bus interconnection) συνδεδεμένοι μεταξύ τους με 10G ethernet). Στους argo-c0-10 θα εκτελούμε MPI, OpenMP και υβριδικά MPI+OpenMp προγράμματα, που είναι το αντικείμενο του παρόντος οδηγού. Ο argo-c10 είναι διαφορετικός, Dell Precision 7920 Rack (Intel Xeon Silver 4114 στα 2.2 GHz, 10 Cores with hyperthreading, 16GB (2X8GB) DDR4 2666MHz με Dual Nvidia Quatro P4000, 1792 Πυρήνες CUDA, RAM 8GB, εύρος ζώνης μνήμης: 243 GB/s) στον οποίο θα τρέχουμε CUDA και OpenMp+CUDA προγράμματα. Οδηγίες σε άλλο κείμενο. Το cluster είναι διαθέσιμο για Δοκιμές, Ανάπτυξη, Debugging, Μετρήσεις, κλπ. στους φοιτητές του μαθήματος και ερευνητές του τμήματος, δεν έχει περιορισμούς, χρονικούς, διαθεσιμότητας πόρων, κλπ. Παρακαλώ, όμως όχι καταχρήσεις, οι πόροι είναι για όλους. Για να συνεχίσετε με τις οδηγίες χρήσης θα χρειαστείτε κλειδί RSA και VPN πρόσβαση. Οδηγίες σε σχετικά αρχεία στο eclass.

Βήμα 1: Πιστοποίηση Χρήστη (Authorization) απαιτείται VPN

1. Με Browser connect to Web Address `https://argo-rbs.cloud.iasa.gr`
2. Επιλογή (Not a Member) Register
3. Στην επόμενη οθόνη εισάγετε τα στοιχεία σας
    a. Full Name (ονοματεπώνυμο με **Αγγλικούς** Χαρακτήρες)
    b. User Name (για το μελλοντικό σας login)
    c. Email (για επικοινωνία έγκρισης)
    d. Password+Confirmation
    e. CAPTCHA.
4. Σημειώστε το Password που βάλατε
5. Πατήστε Register
6. Άμεσα λαμβάνετε email από Argo-Resources Booking System - Registration Confirmation (it@iasa.gr) με οδηγίες ενεργοποίησης του λογαριασμού.
7. Ενεργοποιείτε

Βήμα 2: Υποβολή αιτήματος χρήσης (Challenge Request) απαιτείται VPN

1. Μετά την ενεργοποίηση κάνετε login στην Web Page `https://argo-rbs.cloud.iasa.gr` (username+password από Βήμα1)
2. Επιλέγετε “subscribe to challenges”
3. Βγαίνει User::Info με οδηγίες
4. Upload Public key (copy paste all contents of public key file - text) and submit. **Προσοχή**, επειδή ακόμα δεν υπάρχει αυτόματο sanitization trigger στη βάση, να κάνετε paste το public key σε μια γραμμή, χωρίς line breaks, ειδικά αν χρησιμοποιείτε Windows (λόγω του χαρακτήρα LF στο τέλος της γραμμής).
5. Click “subscriptions section”
6. Make request to Subscribe to available challenges (σε αυτό που ισχύει για το τρέχον ακαδημαϊκό έτος)

7. Θα λάβετε email όταν εγκριθεί το αίτημα (request) σας. Για ασφάλεια χρειάζεται την δική μου τυπική «έγκριση», συνήθως την ίδια μέρα. Αν αργήσω μου στέλνετε email.
8. Μετά από το Request Granted σάς δίδεται λογαριασμός της μορφής argoxxx για login στο front-end μηχάνημα με πρόσβαση στους 10 κόμβους με 80 cores και στον 1 κόμβο με 10 cores και 2 gpus για όλο το ακαδημαϊκό έτος.

Τα Βήματα 1, 2 που ολοκληρώσατε γίνονται μόνο μια φορά στην αρχή. Τα υπόλοιπα βήματα περιγράφουν τον συνήθη κύκλο σύνδεσης στην ΑΡΓΩ, μεταγλώττισης και εκτέλεσης mpi και υβριδικών mpi+openmp προγραμμάτων χρησιμοποιώντας προγράμματα επίδειξης στους 10 κόμβους. (Για CUDA και GPU εκτέλεση σε επόμενες οδηγίες)

Βήμα 3: Login to Front-End of Argo (απαραίτητη VPN σύνδεση)
Με τον λογαριασμό (username) argoxxx που σας έχει δοθεί κάνετε login στο front-end της ΑΡΓΩ από κονσόλα του υπολογιστή σας με χρήση του Private Key και Passphrase (χωρίς άλλο password)

a. Από κονσόλα **linux** με την εντολή
   `ssh -i [path_to_your_private_key] argoxxx@argo-rbs.cloud.iasa.gr`
   σας ζητάει να εισάγετε το Passphrase του RSA.

b. Από **Windows** με putty, δυο επιλογές
   1. Στο Run command (Wind+R) δίνουμε την εντολή
      `putty.exe -ssh -i [path_to_your_private_key] argoxxx@argo-rbs.cloud.iasa.gr`
      Ανοίγει κονσόλα στον front-end, αναγνωρίζει το username και ζητάει Passphrase του RSA.
   2. Τρέχετε το pageant (putty authentication agent), εμφανίζεται εικόνα στο task bar. Ανοίγετε, εισάγετε την θέση του private key το passphrase. Ανοίγετε το putty (ή και το WinSCP) και με σύνδεση στο `argo-rbs.cloud.iasa.gr` κάνει αυτόματα login και authentication.

Μετά την πρώτη σύνδεση, να αντιγράψετε τον φάκελο MpiDemoN στον λογαριασμό με εκτέλεση της εντολής:
`[user@argo]# cp -r /etc/skel2/MpiDemoN .` (η τελεία σημαντική μέρος της εντολής)
Ο φάκελος περιέχει τα εξής:

1. Δύο MPI προγράμματα το mpi_trapDemo.c και το υβριδικό MPI+OpenMp mpiH_trapDemo.c χρήσιμα για επίδειξη χρήσης. Το πρόγραμμα υπολογίζει προσεγγιστικά το **ορισμένο** ολοκλήρωμα $f(x)=x^2$ στο διάστημα 0..3 με την μέθοδο των τραπεζίων. Μαθηματικό αποτέλεσμα 9.0.

2. Το script mpiPBSscript.sh για την εκτέλεση του μεταγλωττισμένου mpi_trapDemo.c και αντίστοιχα το mpihPBSscript.sh για το μεταγλωττισμένο mpiH_trapDemo.c

3. Το αρχείο Commands.txt αρχείο από όπου μπορείτε να κάνετε copy-paste τις εντολές που ακολουθούν (για την ευκολία σας). Δεν συνιστώ copy-paste των εντολών από το παρόν pdf. Μερικά σύμβολα μοιάζουν αλλά δεν είναι σωστά (π.χ. το "-" από το pdf με το σωστό "-")

Τώρα είστε έτοιμοι να μεταγλωττίσετε (Βήμα 4) και να τρέξετε (Βήμα 5) τα προγράμματα

Βήμα 4. Μεταγλώττιση MPI Προγραμμάτων
Αρχικά για την μεταγλώττιση MPI προγραμμάτων, πρέπει να βεβαιωθείτε ότι η ενότητα (module) OpenMPI είναι φορτωμένη (ενεργοποιημένη). Μπορείτε να δείτε ποιες ενότητες είναι φορτωμένες, με την εντολή: `[user@argo]# module list`

Για φόρτωση εν γένει μιας ενότητας εκτελείτε: `[user@argo]# module load name_of_module`

Ειδικά για την φόρτωση του OpenMpi, εκτελείτε: `[user@argo]# module load openmpi4`

Αφού φορτωθεί το OpenMpi, για να μεταγλωττίσετε ένα πρόγραμμα Mpi, χρησιμοποιείτε τον mpicc compiler, με σύνταξη ίδια με τον gcc. Στο παράδειγμα μας, εκτελείτε το script:
`[user@argo]# mpicc -O3 mpi_trapDemo.c -o mpi_trapDemo.x`
Θα δημιουργηθεί το εκτελέσιμο mpi_trapDemo.x

Για μεταγλώττιση MPI προγραμμάτων με profiling με το mpiP (χρήσιμο για τον εντοπισμό προβλημάτων απόδοσης), φορτώστε αρχικά την μονάδα mpip:
`[user@argo]# module load mpip`

και εκτελείτε:
`mpicc -O3 -g mpi_trapDemo.c -L/opt/mpiP-3.5/lib -lmpiP -lbfd -lunwind -o mpi_trapDemo.x`

Για την μεταγλώττιση OpenMP ή υβριδικών MPI + OpenMp προγραμμάτων (περισσότερα στο 5.2) , χρησιμοποιείτε το flag `-fopenmp`, για παράδειγμα:
`[user@argo]# mpicc -O3 -fopenmp mpiH_trapDemo.c -o mpiH_trapDemo.x`

Βήμα 5. Εκτέλεση MPI και Υβριδικών MPI+OpenMp Προγραμμάτων
Για την εκτέλεση των προγραμμάτων, χρησιμοποιούμε τον διαχειριστή πόρων PBS Professional, ο οποίος λειτουργεί με σύστημα ουράς και χειρίζεται πόρους (ως κρατήσεις δωματίων ξενοδοχείων). Θα καταχωρήσετε (qsub) την δουλειά (job) προς εκτέλεση και θα εκτελεστεί όταν έρθει η σειρά της και υπάρχουν διαθέσιμοι οι πόροι που ζητάει. Στην ουρά εισάγετε το αίτημα με ένα script με κατάληξη .sh, το οποίο περιέχει όλες τις πληροφορίες και παραμέτρους για την εκτέλεση που θέλουμε. Στον φάκελο MpiDemo, υπάρχουν δυο scripts, στα οποία μπορείτε να δείτε μερικές βασικές εντολές του PBS, οι οποίες θα αναλυθούν στο 5.1 και 5.2.

Για να καταχωρήσετε μια εργασία στην ουρά εκτελείτε:`[user@argo]# qsub myPBSScript.sh`

Θα σας δώσει ένα `<int>.argo` (όπου int το jobID) ή μήνυμα λάθους για συντακτικό λάθος στο script. Για να δείτε την κατάσταση της εργασίας σας (ένδειξη Q-σε αναμονή, R-Running) εκτελείτε:`[user@argo]# qstat ή qstat <jobID>`

Αν το jobID δεν υπάρχει στην λίστα, η εργασία έχει ήδη τελειώσει (για μικρά προγράμματα είναι πολύ συνηθισμένο). Σε περίπτωση που θέλετε ή χρειάζεται να ακυρώσετε την εκτέλεση, διαγράφετε με
`[user@argo]# qdel <jobID>`

Όταν η εκτέλεση τελειώσει θα έχουν δημιουργηθούν στον κατάλογο δυο αρχεία:

- `myJob.o<id>` για το output της εκτέλεσης [stdout]
- `myJob.e<id>` για τα λάθη εκτέλεσης [stderr]

Όπου το “myJob” ορίζεται από τον χρήστη στο script ρητά ή έμμεσα (δείτε 5.1).

Αν έχετε μεταγλωττίσει με profiling mpiP θα δημιουργηθεί και ένα αρχείο με όνομα της μορφής `mpi_trapDemo.x.32.43450.1.mpiP.` Περισσότερες πληροφορίες στο αντίστοιχο αρχείο.

Ακολουθεί ανάλυση για την Δομή και Χρήση του PBS script για καθαρά MPI προγράμματα (5.1) και υβριδικά MPI + OpenMp (5.2).

5.1 Η Δομή του PBS Script για καθαρά MPI προγράμματα
Με μπλε έντονο χρώμα παρουσιάζονται οι τιμές που μπορείτε να αλλάξετε κατά περίπτωση. Η σειρά εμφάνισης των εντολών δεν είναι σημαντική.

#!/bin/bash

Job Name – βοηθάει για να ξεχωρίζετε κάθε job με κάποιο μνημονικό όνομα#
#PBS -N **myJob**

#Queue για MPI και υβριδικά MPI+OpenMp στους 10 κόμβους (δεν αλλάζει)#
#PBS -q N10C80

Stdout Output File, if not provided a <JobName>.o<JobID> will be created
#PBS -o **job.out**

Stderr Error File, if not provided a <JobName>.e<JobID> will be created
#PBS -e **job.err**

#Change Working directory to SUBMIT directory. THIS IS MANDATORY, PBS Starts everything from $HOME, one should change to submit directory
#cd $PBS_O_WORKDIR

#We next specify resources needed, Memory, Time, Nodes, Cores, MPI Processes

#**Memory** Max VM size, Example, 2GB per process – max 16 GB per node#
#PBS -l pvmem=**2G**

Time Max Wall time Ο χρόνος που θέλετε να δεσμεύσετε του πόρους (κόμβους, πυρήνες)
#PBS -l walltime=**00:01:00** # Example, 1 minute

Reserve (select) Number of **Nodes** and **cpus**/Node. Slots **mpiprocs**/Node,
#Example: reserve 2 nodes and 8 cpus/node running 32 mpi processes (16 per node, 2 per core)
#PBS -l select=**2**:ncpus=**8**:mpiprocs=**16**

#Προσοχή στα Όρια -ARGO select= 1..10;ncpus=1..8; mpiprocs=1..<any>
Αν ζητήσετε πάνω από 10 κόμβους ή πάνω από 8 ncpus/κόμβο το αίτημα θα παραμείνει μόνιμα στην ουρά N10C80, αφού δεν υπάρχουν τόσοι πόροι και επομένως το PBS δεν μπορεί να τους διαθέσει. Το διαγράφετε από την ουρά με qdel <JobID>. Αν το job που δώσατε και με το qstat βλέπετε ότι παραμένει σε κατάσταση αναμονής (Q) χωρίς να υπάρχει άλλο job που τρέχει (R) σημαίνει ότι υπάρχει λάθος στο PBS script. Δεν καταναλώνει πόρους του συστήματος και δεν επηρεάζει την συμπεριφορά των προγραμμάτων που εκτελούνται, απλά δεν μπορεί να εκτελεστεί. Διαγράφετε το job (qdel) και ελέγχετε/διορθώνετε το script πριν το ξανακαταθέσετε (qsub).

Η επόμενη εντολή ορίζει την αποκλειστική χρήση του κόμβου από την εργασία σας. Όταν θα τρέξει θα είναι η μόνη που χρησιμοποιεί τους κόμβους, τους πυρήνες και την μνήμη τους (όχι όμως και το δίκτυο που συνδέει τους κόμβους) Υπάρχουν και άλλες επιλογές που δεν συνιστώνται για την ακεραιότητα των μετρήσεων σας. Η επιλογή αυτή είναι και το default που έχουμε ορίσει για την ουρά N10C80 και μπορεί να απαλειφθεί.

#PBS -l place=excl #Only this job uses the chosen nodes

Runs the compiled program με το mpirun script με τους πόρους που έχουμε ζητήσει

mpirun **mpi_trapDemo.x**
#Here it Runs 32 processes σε 2 κόμβους και 16 πυρήνες, όπως έχουμε ορίσει πριν με (select=**2**:ncpus=**8**:mpiprocs=**16**)

Άλλα Παραδείγματα για παραμετροποίηση εκτέλεσης:

Π1. Για την δημιουργία 16 MPI διεργασιών σε 2 κόμβους των 8 πυρήνων (ncpus) και 8 διεργασίες/κόμβο, ορίζουμε:

#PBS -l select=2:ncpus=8: mpiprocs=8
mpirun <εκτελέσιμο>

Π2. Για την δημιουργία λιγότερων από 8 διεργασίες σε έναν κόμβο π.χ. 1,2,4 (θα το χρειαστούμε για την μελέτη κλιμάκωσης), δεσμεύουμε όλους τους πυρήνες (ώστε να μην τρέξει άλλο πρόγραμμα στον ίδιο κόμβο) και ορίζουμε mpiprocs < 8.
#PBS -l select=2:ncpus=8: mpiprocs=4
mpirun <εκτελέσιμο>

Π3. Για την δημιουργία Μ διεργασιών μη πολλαπλάσιων του 8 (M mod 8 ≠ 0), π.χ. τετράγωνα 9, 25, 36, 49 κλπ. δεσμεύετε select=N (όπου N= ⌊M/8⌋ -round-up), διεργασίες/κόμβο mpiprocs=P (όπου P=⌊M/N⌋ -round-up) και ncpus=8 (συνιστάται πάντα 8-και όχι P) και με την οδηγία
#PBS -l select=N:ncpus=8:mpiprocs=P
ΑΛΛΑ εκτελούμε το mpirun με το switch –np <number of processes>
mpirun -np M <εκτελέσιμο>
Αν εκτελούσαμε χωρίς το -np M θα δημιουργούσαμε NX8XP διεργασίες και όχι Μ.

Για 25 διεργασίες δεσμεύουμε select=⌊25/8⌋= ⌊3.125⌋=4 κόμβους και mpiprocs=⌊25/4⌋= ⌊6.25⌋= 7
#PBS -l **select=4**:ncpus=8:**mpiprocs=7**, ώστε να δώσει την κατανομή [7,7,7,4]
mpirun -np 25 <εκτελέσιμο>]

Επίσης θα δουλέψει και το ακόλουθο (δεν ορίζει όμως το καλύτερο load balancing)
#PBS -l select=4:ncpus=8:**mpiprocs=8**
Αλλά θα κάνει πιο άνιση κατανομή διεργασιών [8,8,8,1] στους κόμβους

Για 36 διεργασίες ⇒ 5 κόμβους, ⌊36/8⌋=⌊4.5⌋=5, N=5 κατανομή [8,8,8,8,4]
#PBS -l select=5:ncpus=8:mpiprocs=8
mpirun -np 36 <εκτελέσιμο>

Συνιστάται να δεσμεύετε πάντα ncpus=8 και να κατανέμετε τις διεργασίες στους κόμβους με το mpiprocs.

5.2 Προσαρμογή PBS script για Υβριδικά προγράμματα MPI + OpenMp
Για υβριδικά προγράμματα χρειάζεται εν γένει 1) να δημιουργούμε λιγότερες από 8 διεργασίες σε κάθε κόμβο 2) να δημιουργούμε νήματα σε κάθε διεργασία και 3) να συνδυάσουμε το 1+2.

1. Για να δημιουργήσουμε λιγότερες από 8 MPI processes/node, έστω P σε κάθε έναν από N nodes, δηλαδή συνολικά P * N διεργασίες:

#PBS -l select=N:ncpus=8:mpiprocs=P #όπου P=1,2,4 (επιτρέπεται και περισσότερες από 8)

π.χ 4 κόμβους και 2 διεργασίες/κόμβο με δέσμευση όλων (8) των cpus

#PBS -l select=4:ncpus=8:mpiprocs=2
mpirun --bind-to none <εκτελέσιμο>

Προσοχή: Με απλό mpirun όλες οι διεργασίες τρέχουν στον έναν από τους δυο επεξεργαστές των κόμβων (στους 4 πυρήνες) ενώ ο άλλος είναι αδρανής.

2. Έχοντας ορίσει αριθμό διεργασιών ανά κόμβο μπορούμε να ορίσουμε στο PBS script τον προεπιλεγμένο (default) αριθμό νημάτων Τ που δημιουργεί κάθε διεργασία με:

#PBS -l select=N:ncpus=8:mpiprocs=P:**ompthreads=T**

Ο Τ μπορεί να είναι Τ ≥ 1 με ανώτατο όριο εξαρτώμενο από την αρχιτεκτονική, τυπικά κοντά στο (υπερβολικό) 109. Πρακτικά εμείς θα θεωρήσουμε Τ= (συνήθως) 2, 4, το πολύ 8 νήματα ανά διεργασία. Αν δεν ορίσουμε το ompthreads θα πάρει ompthreads=ncpus

Διευκρινίζουμε ότι με το ompthreads ορίζουμε μόνο τον προεπιλεγμένο (default) αριθμό νημάτων, που δημιουργεί ένα πρόγραμμα και μπορούμε να το διαπιστώσουμε στο OpenMp πρόγραμμα με int omp_get_num_threads(void);

Φυσικά μπορούμε σε OpenMp πρόγραμμα να αλλάξουμε τον «προεπιλεγμένο» αυτόν αριθμό με: void omp_set_num_threads(int num_threads);

ή ακόμα να τον ορίσουμε προσωρινά με την ιδιότητα (attribute) num_threads (Τ1), π.χ.
#pragma omp parallel num_threads(3)
το οποίο θα αγνοήσει προσωρινά την τιμή Τ που ορίσατε στο script

Επομένως το ompthreads είναι μόνο για ευκολία μας, ώστε να έχουμε λιγότερες παραμέτρους στο πρόγραμμά μας ή και να αποφεύγουμε μεταγλώττιση λόγω αλλαγής αριθμών νημάτων στις μετρήσεις. ΔΕΝ δημιουργούνται νήματα, μόνο οι διεργασίες! Τα νήματα δημιουργούνται από την (υβδιδική) mpi διεργασία εσωτερικά.

3. Για να τρέξετε ένα υβριδικό πρόγραμμα MPI+OpenMp συνδυάζετε το 1) την δημιουργία διεργασιών ανά κόμβο και το 2) τον ορισμό του προκαθορισμένου αριθμού νημάτων ανά διεργασία. Για να δημιουργήσετε P διεργασίες ανά κόμβο και Τ νήματα ανά διεργασία σε Ν κόμβους με 8 cpus ορίζετε:

#PBS -l select=N:ncpus=8:mpiprocs=P:ompthreads=T
mpirun --bind-to none <εκτελέσιμο>

Παράδειγμα 1: Για 3 κόμβους, 8 πυρήνες/κόμβο, 2 διεργασίες/κόμβο και 4 νήματα/διεργασία

#PBS -l select=3:ncpus=8:mpiprocs=2:ompthreads= 4
mpirun --bind-to none <εκτελέσιμο>

Παράδειγμα 2: Για να ζητήσετε 2 κόμβους, 8 πυρήνες/κόμβο και 8 διεργασίες/κόμβο με 4 νήματα/διεργασία:

#PBS -l select=2:ncpus=8:mpiprocs=8:ompthreads=4

Παράδειγμα 3: Για 25 διεργασίες και 2 νήματα/διεργασία για να ισο-κατανείμουμε τις διεργασίες σε κόμβους και 1 νήμα/cpu [4,4,4,4,4,4,1] δεσμεύουμε κόμβους των 8 cpu (⌊(25*2)/8⌋= ⌊6.25⌋=7), select N=7 και mpiprocs=4 (max πόρους για 7*4=28 διεργασίες)
#PBS -l select=7:ncpus=8:mpiprocs=4:ompthreads=2
mpirun --bind-to none -np 25 <εκτελέσιμο>]

Σημειώνουμε ότι οι παράμετροι του PBS select,ncpus, mpiprocs, ompthreads εντάσσονται στην 4^{η} φάση της μεθοδολογίας του Foster (mapping)