

Γενική Χρήση του ARIS

Δρ. Δημήτρης Ντελλής

GRNET

ntell [at] grnet.gr

Γενική Χρήση
του ARIS

Δρ. Δημήτρης
Ντελλής

Περιεχόμενα

Environment
Modules

Χρήση των modules

Batch System

SLURM

Εντολές SLURM

SLURM Limits

PBS emulation

Accounting

Διαθέσιμα
πακέτα

Βέλτιστες
Πρακτικές -
Κοινά
προβλήματα

Πρακτική
εξάσκηση

Περιεχόμενα

- Software Environment
 - Environment Modules
 - Διαθέσιμα πακέτα
- Batch System
 - Job Submission
 - Job Control
 - Accounting
- Βέλτιστες Πρακτικές Χρήσης - Συνηθισμένα Λάθη/Προβλήματα.
- Πρακτική εξάσκηση, παραδείγματα

Environment Modules. Τι είναι ?

- Το πακέτο Environment Modules κάνει δυναμική τροποποίηση του περιβάλλοντος χρήστη μέσω των module files.
- Κύριες μεταβλητές περιβάλλοντος που προσαρμόζονται είναι οι PATH, MANPATH, και LD_LIBRARY_PATH, αλλά και μεταβλητές περιβάλλοντος που ενδεχομένως κάθε πακέτο λογισμικού χρειάζεται.
- Κάθε module file περιέχει την πληροφορία που χρειάζεται ώστε να ρυθμίσει τις μεταβλητές περιβάλλοντος για κάποια εφαρμογή.

- Όλα τα modules θέτουν μια μεταβλητή `MODULENAMEROOT`. Σε modules που αναφέρονται σε βιβλιοθήκες, συνήθως τα `include files` βρίσκονται στην `$MODULENAMEROOT/include` και οι βιβλιοθήκες στην `$MODULENAMEROOT/lib`
- Εάν υπάρχουν εξαρτήσεις ενός πακέτου λογισμικού από άλλα τα οποία επίσης ρυθμίζονται με `module file`, οι εξαρτήσεις αυτές μπορούν να περιγραφούν και εφόσον το αντίστοιχο `module` δεν είναι ενεργό είτε το φορτώνει είτε βγάζει μήνυμα λάθους ειδοποιώντας το χρήστη ότι πρέπει πρώτα να φορτώσει τις εξαρτήσεις.
- Σε περιπτώσεις πακέτων τα οποία υπάρχουν σε πάνω από μια έκδοση, υπάρχει ένα `module` για κάθε έκδοση και ο `administrator` μπορεί να ορίσει κάποια ως `default`.

Environment Modules. Χρήση

- Έλεγχος πακέτων που είναι διαθέσιμα μέσω modules
`module avail`
ή
`module -l avail`
- Έλεγχος ενεργών modules
`module list`
- Απενεργοποίηση όλων των ενεργών modules
`module purge`
- Απενεργοποίηση συγκεκριμένου module
`module unload MODULENAME`

- Αλλαγή έκδοσης module

```
module switch MODULENAME/VER1 MODULENAME/VER2
```

- Πληροφορίες για το τι αφορά κάποιο module

```
module whatis MODULENAME/VERSION
```

- Κείμενο Βοήθειας για κάποιο module

```
module help MODULENAME/VERSION
```

- Για να δείτε τι κάνει η ενεργοποίηση ενός module

```
module show MODULENAME/VERSION
```

- Default version ενός module
 - Όπως θα δείτε παρακάτω, σχεδόν όλα τα πακέτα που υπάρχουν στο ARIS σε πάνω από μια version έχουν μια από αυτές επισημασμένη ως default. Στην περίπτωση αυτή, οι εντολές
`module load MODULENAME`
και
`module load MODULENAME/DEFAULTVERSION`
είναι ισοδύναμες.

Batch System

- Τι είναι ένα Batch System
 - Ένα Batch System ελέγχει την πρόσβαση στους διαθέσιμους υπολογιστικούς πόρους ώστε όλοι οι χρήστες να μπορούν να χρησιμοποιούν το σύστημα - Συνήθως σε ένα σύστημα υπάρχει μεγαλύτερη ζήτηση για πόρους από τους διαθέσιμους.
 - Δίνει τη δυνατότητα στο χρήστη να προδιαγράψει μια υπολογιστική εργασία (Job) , να την υποβάλει στο σύστημα και να αποσυνδεθεί από αυτό.
 - Η εργασία θα εκτελεστεί όταν υπάρχουν πόροι (cores, nodes, μνήμη) και χρόνος
- ARIS Batch System : SLURM, υποστηρίζεται PBS emulation

Όταν μια εργασία υποβάλεται σε ένα Batch system :

- Περιγράφονται οι πόροι που χρειάζεται το σύστημα (π.χ. cores, nodes, μνήμη, χρόνος εκτέλεσης)
- Το σύστημα κατάγράφει τους πόρους που ζητήθηκαν
- Όταν βρεθούν οι διαθέσιμοι πόροι, ξεκινάει η εκτέλεση της εργασίας
- Οι πόροι μπορούν να χρησιμοποιηθούν όπως θέλει ο χρήστης

SLURM Scripts

Ένα SLURM Script περιγράφει τους πόρους που χρειάζεται για να τρέξει η εργασία, όπως επίσης τις εντολές εκτέλεσης της εργασίας.

Παρατηρήστε τους 2 τρόπους που μπορούν να περιγραφούν οι απαιτήσεις της εργασίας π.χ.

```
--nodes=200
```

και

```
-A sept2015
```

.

SLURM Scripts

```
#!/bin/bash
#SBATCH --job-name="testSlurm" # Όνομα για διαχωρισμό μεταξύ jobs
#SBATCH --error=job.err.%j # Filename για το stderr
#SBATCH --output=job.out.%j # Filename για το stdout
# Το %j παίρνει την τιμή του JobID
#SBATCH --nodes=200 # Αριθμός nodes
#SBATCH --ntasks=400 # Αριθμός MPI Tasks
#SBATCH --ntasks-per-node=2 # Αριθμός MPI Tasks / node
#SBATCH --cpus-per-task=10 # Αριθμός Threads / MPI Task
#SBATCH --mem=56G # Μνήμη ανά node
#SBATCH --mem-per-cpu=2800M # Μνήμη ανά core
#SBATCH -A sept2015 # Accounting tag (sept2015 για
# όλους στο training)
#SBATCH -t 01:00:00 # Ζητούμενος χρόνος HH:MM:SS
#SBATCH -p compute # partition, default στο ARIS.

module purge
module load gnu
module load intel
module load intelmpi
export OMP_NUM_THREADS=$SLURM_CPUS_PER_TASK

srun EXECUTABLE ARGUMENTS
```

SLURM Scripts

- Το script του προηγούμενου slide είναι η πλήρης περιγραφή μιας εργασίας.
- Μπορεί να υποβληθεί εργασία και με λιγότερα από τα #SBATCH directives
 - Δίνοντας μόνο το `--nodes` χωρίς το `--ntasks` το σύστημα μπορεί να υπολογίσει πόσα tasks θα χρησιμοποιήσει
 - Αντίστοιχα, δίνοντας μόνο το `--ntasks` το σύστημα μπορεί να υπολογίσει πόσα nodes χρειάζεται.
 - Τα υποχρεωτικά που σχετίζονται με τον αριθμό των cores που θα χρησιμοποιήσει μια εργασία είναι ένα από τα παραπάνω
 - Παραλείποντας το `--job-name`, το σύστημα το θέτει ίδιο με το όνομα του script.
 - Παραλείποντας το `--output` το σύστημα το θέτει σε `slurm-JOB_ID.out`
 - Υποχρεωτική είναι η χρήση του `--account` (ή `-A`)
 - Θέτοντας όλες τις μεταβλητές έχετε πλήρη έλεγχο του τι πόρους ζητάτε από το σύστημα.

Χρήση **srun** για την εκτέλεση των εφαρμογών

- Οι εκδόσεις του MPI έχουν η κάθε μια ένα `mpirun`/`mpiexec` κλπ.
- Προτείνεται να χρησιμοποιείται το `srun` για την εκτέλεση παράλληλων εργασιών.
- Κάποιοι από τους λόγους
 - Το `srun` ξεκινάει τα εκτελέσιμα σε όλους τους κόμβους οπότε έχει πλήρη έλεγχο.
 - Το `srun` κάνει accounting κατανάλωσης ρεύματος, χρήση Infiniband, χρήση δίσκων, κλπ.
 - Είναι κοινός τρόπος για τις (3 προς στιγμήν) εκδόσεις MPI που υπάρχουν στο ARIS
 - Η χρήση `mpirun`, `mpiexec` κλπ. δεν συνιστάται. Σε περιπτώσεις που η εφαρμογή έχει προβλήματα και σταματήσει σταματήσει ίσως να παρουσιαστούν προβλήματα (zombie procs) στη χρήση του `scancel`.

Επικοινωνία με το SLURM

- Υποβολή εργασίας
`sbatch SLURM_JobScript.sh`
Submitted batch job 15242
- Κατάλογος εργασιών
`squeue`
- Κατάλογος εργασιών με περισσότερες λεπτομέρειες
`squeue -o "%.8i %.9P %.10j %.10u %.8T %.5C
%.4D %.6m %.10l %.10M %.10L %.16R"`
- Ακύρωση εργασίας
`scancel JobID`
- Σε κάποιες περιπτώσεις που τα εκτελέσιμα δεν
τερματίζονται άμεσα παίρνοντας SIGHUP από το
SLURM
`scancel -s KILL JobID`
- Εκτίμηση του πότε θα αρχίσει η εκτέλεση των εργασιών
που είναι σε αναμονή για πόρους
`squeue --start`
- Πληροφορίες για την τρέχουσα χρήση
`sinfo`

SLURM jobs dependency

- Εάν μια εργασία για να αρχίσει πρέπει κάποια άλλη να έχει ήδη αρχίσει ή τελειώσει, στο SLURM Script εκτός των άλλων :

```
#SBATCH --dependency=after:Job_ID
```

ή

```
#SBATCH --dependency=afterok:Job_ID
```

αντίστοιχα

- Εάν μια εργασία για να αρχίσει πρέπει κάποια άλλη με το ίδιο job name και χρήστη να έχει τελειώσει, στο SLURM Script εκτός των άλλων :

```
#SBATCH --dependency=singleton
```

- Εάν πρέπει μια εργασία να ξεκινήσει κάποιο συγκεκριμένο χρονικό διάστημα, στο SLURM Script εκτός των άλλων :
 - Έναρξη στις 16:00
`#SBATCH --begin=16:00`
 - Έναρξη συγκεκριμένη ημέρα και ώρα :
`#SBATCH --begin=2015-12-17T14:32:00`

Εάν κάποια εργασία δεν τρέχει και στο nodelist/REASON εμφανίζονται τιμές εκτός από nodenames ή Resources, τότε έχουμε ζητήσει περισσότερους πόρους από ότι μας επιτρέπεται

- `AssocMaxNodesPerJobLimit`
Ζητάμε περισσότερα nodes από ότι επιτρέπεται στο account μας
- `AssocMaxWallDur`
Ζητάμε περισσότερο χρόνο από ότι επιτρέπεται στο account μας
- Διάφοροι άλλοι λόγοι που εάν από το όνομα δεν είναι αντιληπτό, ανατρέξτε στο documentation του SLURM.

SLURM Environment Variables

Όταν ξεκινάει η εργασία το SLURM βάζει κάποιες μεταβλητές που σχετίζονται με αυτή, και ενδεχομένως είναι χρήσιμες στον χρήστη.

```
$SLURM_NNODES          # Αριθμός nodes
$SLURM_NTASKS          # Αριθμός Tasks
$SLURM_NPROCS          # " " "
$SLURM_NTASKS_PER_NODE # Αριθμός Tasks /node
$SLURM_TASKS_PER_NODE  # " " " "
$SLURM_CPUS_PER_TASK   # Αριθμός threads / Task
$SLURM_MEM_PER_NODE    # Μνήμη / node (MB)
```

SLURM User/Group resource limits

- Στο SLURM το κάθε account έχει κάποια όρια πόρων που μπορεί να ζητήσει/χρησιμοποιήσει. Τα όρια αυτά εφαρμόζονται σε όλους του χρήστες του account. Αυτά είναι :
 - Αριθμός Jobs που μπορούν να εκτελούνται ταυτόχρονα
 - Αριθμός Jobs που μπορούν να εκτελούνται ή να βρίσκονται σε αναμονή
 - Μέγιστος αριθμός cores ή nodes που μπορούν να χρησιμοποιηθούν ταυτόχρονα από jobs ενός account.
 - Μέγιστη χρονική διάρκεια εκτέλεσης ενός Job
 - Μέγιστος αριθμός nodes που μπορεί να ζητήσει ένα Job
 - Μέγιστος αριθμός cores που μπορεί να ζητήσει ένα Job
 - Συνολικός αριθμός core hours στη διάρκεια ενός project.

- Ο Scheduler στο ARIS είναι FIFO with Backfill. Αυτό σημαίνει
 - Το job που υποβλήθηκε πρώτο θα εκτελεστεί πρώτο
 - Από τη στιγμή που ξεκινάει η εκτέλεση, η εργασία θα τελειώσει το αργότερο μετά από όσο χρόνο ζητήθηκε στο SLURM script.
 - Εάν το σύστημα έχει μεν ελευθερους πόρους (cores/nodes/memory) αλλά δεν είναι αρκετοί για να τρέξει το πρώτο στη σειρά από τα queued, τα επόμενα jobs θα περιμένουν
 - εκτός...

- Κάποιο από τα επόμενα jobs ζητάει πόρους που υπάρχουν, και ο χρόνος εκτέλεσης που ζητάει είναι μικρότερος από τον πιο κοντινό αναμενόμενο χρόνο τέλους των jobs που εκτελούνται. Αυτό το job θα παρακάμψει τη σειρά, και θα εκτελεστεί πρώτο χωρίς να προκαλέσει καμιά καθυστέρηση σε άλλα jobs.
- Έτσι το σύστημα έχει τη μεγαλύτερη δυνατή χρήση.
- Ζητήστε λίγο παραπάνω από όσο χρόνο υπολογίζετε ότι χρειάζεται η εργασία σας και όχι το μέγιστο που μπορείτε

Εξομοίωση PBS

- Υπάρχει εγκατεστημένη η εξομοίωση του PBS/Torque. Χρήστες που είναι εξοικωμένοι στη χρήση PBS μπορούν να χρησιμοποιήσουν τα PBS scripts και εντολές.
- Η εξομοίωση του PBS καλύπτει μεγάλο βαθμό περιγραφής εργασιών, αλλά όχι όλα

SLURM

sbatch
squeue
scancel

```
#!/bin/sh
#SBATCH --mem-per-cpu=2G
#SBATCH -t 1:00:00
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=20
#SBATCH -A sept2015
#SBATCH -p compute
....
```

PBS

qsub
qstat
qdel

```
#!/bin/sh
#PBS -l pvmem=2G
#PBS -l walltime=1:00:00
#PBS -l nodes=1:ppn=20
#PBS -A sept2015
#PBS -q compute
....
```

Accounting

- Δείτε τα jobs σας το τρέχον 24ωρο
sacct
- Δείτε τα jobs σας τον τελευταίο μήνα
sacct -S 2015-08-14
- Δείτε πόσο χρόνο (και ενέργεια σε Wh εφόσον χρησιμοποιείτε srun) έχετε καταναλώσει το τελευταίο εξάμηνο
myreport
- Δείτε πόσο από τον χρόνο που σας έχει δοθεί έχετε χρησιμοποιήσει
mybudget

Γενική Χρήση
του ARIS

Δρ. Δημήτρης
Ντελλής

Περιεχόμενα

Environment
Modules

Χρήση των modules

Batch System

SLURM

Εντολές SLURM

SLURM Limits

PBS emulation

Accounting

Διαθέσιμα
πακέτα

Βέλτιστες
Πρακτικές -
Κοινά
προβλήματα

Πρακτική
εξάσκηση

Διαθέσιμα πακέτα

- Compilers/Debuggers
- MPI Implementations
- Libraries
- Applications
- Debuggers/Profilers
- Graphics
- Εφαρμογές

Compilers

- Εγκατεστημένοι Compilers
 - Intel 15.0.3 (default), 16.0.0
 - module load intel (intel/16.0.0)
 - icc, icpc, ifort
 - Βασικά Flags : -O3 -xCORE-AVX-I (-xAVX)
 - OpenMP : -openmp
 - GNU 4.9.2 (default), 4.9.3, 5.1.0, 5.2.0
 - module load gnu (gnu/4.9.3, κλπ.)
 - gcc, g++, gfortran
 - Βασικά Flags : -O3 -mavx -march=ivybridge -mtune=ivybridge
 - OpenMP : -fopenmp

Γενική Χρήση
του ARIS

Δρ. Δημήτρης
Ντελλής

Περιεχόμενα

Environment
Modules

Χρήση των modules

Batch System

SLURM

Εντολές SLURM

SLURM Limits

PBS emulation

Accounting

Διαθέσιμα
πακέτα

Βέλτιστες
Πρακτικές -
Κοινά
προβλήματα

Πρακτική
εξάσκηση

Debuggers

- gdb 7.9.1
- Intel gdb 15.0.3, 16.0.0
- ddd

MPI

- Intel MPI 5.0.3 (default), 5.1.1
- OpenMPI 1.8.8, 1.10.0, for GNU and Intel
- MVAPICH2 2.2.2a (experimental) for GNU και Intel

Σημειώσεις για τον IntelMPI

- Οι wrappers **mpicc/mpicxx/mpif90** του IntelMPI χρησιμοποιούν GNU compilers
- Υπάρχουν οι αντίστοιχοι wrappers (και headers/libraries) για Intel Compilers **mpiicc/mpiicpc/mpiifort**.

Σημειώσεις για τον MVAPICH2

- Η χρήση του mvarich2 υποστηρίζεται ΜΟΝΟ μέσω του **srun** => δεν υπάρχει mpirun, mpiexec κλπ.
- Με οποια από τις versions (gnu/intel)
- **mpicc -cc=icc, mpicxx -cxx=icpc, mpif90 -fc=ifort**

MPI

Εκτέλεση MPI εφαρμογών

- Οι εκδόσεις του MPI έχουν η κάθε μια ένα mpiun/mpiexec κλπ.
- Προτείνεται να χρησιμοποιείται το srun για την εκτέλεση παράλληλων εργασιών.
- Κάποιοι από τους λόγους
 - Το srun ξεκινάει τα εκτελέσιμα σε όλους τους κόμβους οπότε έχει πιο πλήρη έλεγχο.
 - Το srun κάνει accounting κατανάλωσης ρεύματος, χρήση Infiniband, χρήση δίσκων, κλπ.
 - Είναι κοινός τρόπος για τις (3 προς στιγμήν) εκδόσεις MPI που υπάρχουν στο ARIS

Γενική Χρήση του ARIS

Δρ. Δημήτρης
Ντελλής

Περιεχόμενα

Environment
Modules

Χρήση των modules

Batch System

SLURM

Εντολές SLURM

SLURM Limits

PBS emulation

Accounting

Διαθέσιμα πακέτα

Βέλτιστες
Πρακτικές -
Κοινά
προβλήματα

Πρακτική
εξάσκηση

- Σε περιπτώσεις που η εφαρμογή έχει προβλήματα και χρειαστεί να σταματήσει ίσως να παρουσιαστούν προβλήματα (zombie procs) στη χρήση του **scancel**, όταν αυτή έχει ξεκινήσει με **mpirun**.
- Η χρήση **mvarich2** υποστηρίζεται (προς στιγμήν) **MONO** με **srun**.

Γενική Χρήση
του ARIS

Δρ. Δημήτρης
Ντελλής

Περιεχόμενα

Environment
Modules

Χρήση των modules

Batch System

SLURM

Εντολές SLURM

SLURM Limits

PBS emulation

Accounting

Διαθέσιμα
πακέτα

Βέλτιστες
Πρακτικές -
Κοινά
προβλήματα

Πρακτική
εξάσκηση

Profilers

- gprof
- mpiP
- Scalasca
- Intel VTune

Βιβλιοθήκες

module avail για να δείτε την τρέχουσα πλήρη λίστα.

```
atlas/3.11.34
boost/1.58.0
cgnslib/3.2.1/intel
elpa/2015.05.001/intel
fftw/2.1.5
fftw/3.3.4/avx
fftw/3.3.4/sse2
flame/5.0/gnu
flame/5.0/intel
glpk/4.55
gsl/1.16/gnu
hdf5/1.8.12/gnu
hdf5/1.8.12/intel
hdf5/1.8.15/gnu
hdf5/1.8.15/intel
jasper/1.900.1
libint/1.1.5
libjpeg-turbo/1.4.1
libsmm/gnu
libsmm/intel
libxc/2.2.2
netcdf/3.6.3/intel
netcdf/4.1.3/gnu
netcdf/4.1.3/intel
netcdf-c/4.3.3.1/gnu
netcdf-c/4.3.3.1/intel
netcdf-combined/4.3.3.1/intel
netcdf-fortran/4.4.2/gnu
netcdf-fortran/4.4.2/intel
openblas/0.2.14/gnu/int4
openblas/0.2.14/gnu/int8
openblas/0.2.14/intel/int4
openblas/0.2.14/intel/int8
parmetis/4.0.3/gnu
parmetis/4.0.3/intel
pnetcdf/1.6.1/gnu
pnetcdf/1.6.1/intel
scalapack/2.0.2/gnu
scalapack/2.0.2/intel
szip/2.1
udunits2/2.2.19
voro++/0.4.6
```


Εφαρμογές

```
abinit/7.10.4
bigdft/1.7.6
cdo/1.7.0
code_saturne/4.0.1/intel
cp2k/2.6.1
cpmd/4.1
dlpoly/2.20
dlpoly/4.07
gamess-US/2014R1
gopenmol/3.00
gromacs/4.5.7
gromacs/4.6.7
gromacs/5.0.5
gromacs/5.0.6
gromacs/5.1
lammps/15May15
mdynamix/5.2.7
molden/5.2
molekel/5.4.0
mpqc/2.3.1
namd/2.10/hybrid/memopt
namd/2.10/hybrid/normal
namd/2.10/purempi/memopt
namd/2.10/purempi/normal
ncarg/6.3.0
ncview/2.1.5
nwchem/6.5
octave/4.0.0
octopus/4.1.2
openbabel/2.3.2
openmd/2.2
paraview/4.3
qhull/2012.1
quantum-espresso/5.2.0
R/3.2.1
towhee/7.1.0
vmd/1.9.2
wrf/3.4.1/hybrid
wrf/3.4.1/purempi
wrf/3.7/hybrid
wrf/3.7/purempi
wrf-chem/3.7
wrf-chem/3.7-hybrid
```

Εφαρμογές

- Εκτός από τα πακέτα που είναι διαθέσιμα μέσω modules υπάρχουν και πακέτα που ενδεχομένως είναι χρήσιμα από το σύστημα.
 - Gnuplot
 - Grace
 - Gimp

Βέλτιστες Πρακτικές

- Τα nodes του ARIS διαθέτουν 20 cores και 64 GB RAM Διαθέσιμα για jobs τα 56 GB. Χρησιμοποιήστε πλήρως τα cores των nodes, δηλ. 20 cores/node.

```
--tasks-per-node=20
```

```
--cpus-per-task=1
```

ή

```
--tasks-per-node=2
```

```
--cpus-per-task=10
```

ή άλλους συνδιασμούς tasks/threads με γινόμενο 20.

- Σε περίπτωση που χρειάζεται RAM πάνω από 2.8 GB/core, μπορεί να ζητηθούν λιγότερα cores/node με ταυτόχρονη αύξηση της μνήμης / task, π.χ.

```
--tasks-per-node=18
```

```
--cpus-per-task=1
```

```
--mem-per-task=3.1G
```

Βέλτιστες Πρακτικές

- Σε περίπτωση που οι απαιτήσεις μνήμης δεν είναι ίδιες για όλα τα process, χρησιμοποιήστε τη μεταβλητή για συνολική μνήμη / node.

```
--tasks-per-node=20
```

```
--cpus-per-task=1
```

```
--mem=56G
```

Βέλτιστες Πρακτικές

- Εάν για κάποιον λόγο χρειάζεται αριθμός cores όχι πολλαπλάσιο του 20, συνήθως δυνάμεις του 2 (256, 512, κλπ.)
 - Χρησιμοποιήστε το μικρότερο δυνατό αριθμό nodes.

cores	Nodes	tasks/node	Αχρησιμοποίητα cores
64	4	20	16 σε 1 node
128	7	20	12 σε 1 node
256	13	20	4 σε 1 node
512	26	20	8 σε 1 node

- Σύνηθες λάθος που μεταφέρεται από τη χρήση συστημάτων με 12 ή 16 cores

cores	Nodes	tasks/node	Αχρησιμοποίητα cores
64	4	16	4 cores/node σε 4 nodes = 16
90	6	15	5 cores/node σε 6 nodes = 30
128	8	16	4 cores/node σε 8 nodes = 32
480	40	12	8 cores/node σε 40 nodes = 320
512	32	16	4 cores/node σε 32 nodes = 128

Βέλτιστες Πρακτικές

- Αρκετά πακέτα διαθέτουν ρυθμίσεις για τα όρια μνήμης στο input τους. Φροντίστε να είναι σε συμφωνία με τα όρια μνήμης που ζητούνται από το SLURM.
- Για jobs που έχουν μεγάλο I/O, χρησιμοποιήστε το χώρο σας στην \$WORKDIR.
- Εάν έχετε το δικό σας κώδικα και κάνετε μεταγλώτιση, χρησιμοποιήστε τα κατάλληλα για το σύστημα compiler flags.
- Χρησιμοποιήστε κατά το δυνατόν τις διαθέσιμες Μαθηματικές βιβλιοθήκες που υπάρχουν στο σύστημα και είναι βελτιστοποιημένες για αυτό.

Βέλτιστες Πρακτικές

- Εάν για κάποιο λόγο πρέπει να χρησιμοποιήσετε `mpirun`, χρησιμοποιήστε το χωρίς τα συνήθη `-np`, `-machinefile` κλπ. Συμβαίνει όταν χρησιμοποιούνται, να μην αλλάζει ταυτόχρονα ο αριθμός των `tasks` στο SLURM και ο αριθμός των `tasks` στο `mpirun -np` π.χ.

```
#SBATCH --nodes=10
```

```
#SBATCH --ntasks=200
```

```
mpirun -np 8
```

Δεσμεύετε (και χρεώνεστε) για 200 cores ενώ χρησιμοποιείτε μόλις 8.

Βέλτιστες Πρακτικές

- Εάν η εφαρμογή σας χρησιμοποιεί OpenMP :
 - Φροντίστε ώστε να δίνετε τα σωστά threads/task στο SLURM.
 - Κοινά λάθη :
 - Δεν θέτουμε τη μεταβλητή `OMP_NUM_THREADS=$SLURM_CPUS_PER_TASK`
 - Για όσο χρόνο το job μας τρέχει μόνο του στο node, μπορεί να χρησιμοποιεί όλα τα cores. Εάν έρθει και άλλο job στο node, τότε το load του node θα ανέβει πάνω από 20 και το performance των jobs εξαρτάται κατά πολύ από τα υπόλοιπα jobs στο node.
 - Με Hybrid MPI/OpenMP εφαρμογές, αν δεν θέσουμε τη μεταβλητή `OMP_NUM_THREADS` και χρησιμοποιούμε π.χ. 20 tasks/node, τότε το load του node γίνεται $20 \times 20 = 400$, με αποτέλεσμα ελλειπωμένο performance.

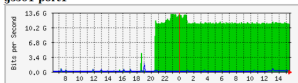
Βέλτιστες Πρακτικές

- Εξερευνήστε την εφαρμογή σας για πιθανές λεπτομέρειες που αφορούν τις επιδόσεις, ειδικά εάν υπάρχει αρκετό I/O.
- Παραδείγματα : quilting στο wrf, Scratch space σε εφαρμογές quantum mechanics.

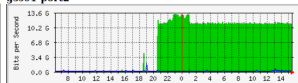
Βέλτιστες Πρακτικές

- Παράδειγμα Βαριάς χρήσης SCRATCH : Διάβασμα από files με ρυθμό 12.6 GBytes/s για 2 ημέρες = 2.12 PBytes για 1 job των 100 cores!!!!.

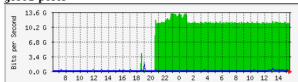
gss01-port1



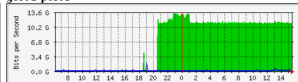
gss01-port2



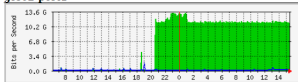
gss01-port3



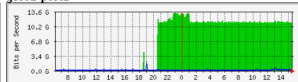
gss01-port4



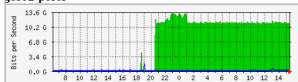
gss02-port1



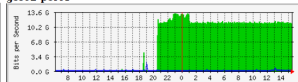
gss02-port2



gss02-port3

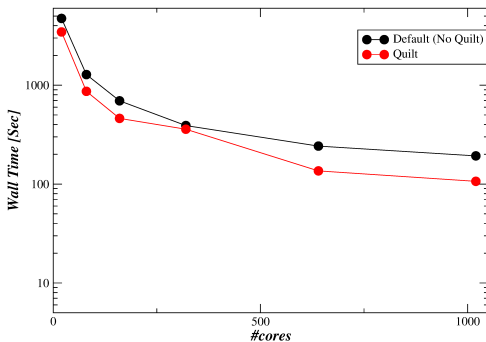


gss02-port4



Βέλτιστες Πρακτικές

- Παράδειγμα performance/scaling WRF με και χωρίς quilting



Βέλτιστες Πρακτικές

WRF: 24ωρη πρόγνωση για Ελλάδα.

#Cores	Time		Core Hours	
	No Quilt	Quilt	No Quilt	Quilt
	[sec]			
20	4733.36	3455.81	26.30	19.20
80	1280.50	866.51	28.46	19.26
160	695.09	461.49	30.89	20.51
320	390.76	358.91	34.73	31.90
640	242.12	136.02	43.04	24.18
1020	192.81	106.58	54.63	30.20

- Χρησιμοποιώντας την τεχνική έχω το αποτέλεσμα που θέλω εντός 136 sec, με κόστος 24.18 core hours αντί 192 sec με κόστος 54 core hours.

Βέλτιστες Πρακτικές

- Σε περίπτωση που η εφαρμογή σας χρησιμοποιεί υβριδικό παραλληλισμό, εξερευνήστε τη συμπεριφορά της με διάφορους συνδιασμούς Tasks/Threads per Task.
- Εάν η εφαρμογή σας έχει διαδικασία save/restart χρησιμοποιήστε τη. Αντί για jobs της π.χ. 1 εβδομάδας, προτιμήστε 7 jobs της 1 ημέρας χρησιμοποιώντας τα dependencies του SLURM.
- Βασικό πρόβλημα στα Hexascale συστήματα.

Βέλτιστες Πρακτικές

- Χρησιμοποιήστε το template στο <http://doc.aris.grnet.gr/scripttemplate/> για να φτιάξετε το SLURM script σας. Περιέχει έλεγχο για το αν θέτετε ή όχι τις αντίστοιχες μεταβλητές.
- Επικοινωνήστε με το [support \[at\] hpc.grnet.gr](mailto:support[at]hpc.grnet.gr) για όποιο πρόβλημα ή απορία έχετε.

Γενική Χρήση
του ARIS

Δρ. Δημήτρης
Ντελλής

Περιεχόμενα

Environment
Modules

Χρήση των modules

Batch System

SLURM

Εντολές SLURM

SLURM Limits

PBS emulation

Accounting

Διαθέσιμα
πακέτα

Βέλτιστες
Πρακτικές -
Κοινά
προβλήματα

Πρακτική
εξάσκηση

Ερωτήσεις ?

Πρακτική εξάσκηση

- Εξοικίωση με το σύστημα
 - Εξερευνήστε τα modules, load/switch/unload
 - Ετοιμάστε SLURM job scripts, δοκιμάστε υποβολή, ακύρωση, εξαρτήσεις και γενικά ότι ταιριάζει στην καθημερινή απασχόλησή σας.
 - Δοκιμάστε λάθη στο SLURM script και δείτε τη συμπεριφορά του συστήματος.
 - **Σημείωση : Επειδή υπάρχει μόνο 2 seats άδεια για Intel Compilers, προτιμήστε τους GNU compilers σε όλες τις πρακτικές προς αποφυγή καθυστερήσεων.**
 - Εφόσον δεν χρησιμοποιείτε/γνωρίζετε κάποια απο τις εφαρμογές που είναι εγκατεστημένες, ή δεν έχετε κάποιο μικρό κώδικα να κάνετε compile, κάνετε δοκιμές με **srun hostname**

Παραδείγματα βέλτιστης χρήσης του συστήματος

- Μαθηματικά προβλήματα : Πολλαπλασιασμός Πινάκων, Επίλυση γραμμικού συστήματος εξισώσεων.
 - Εξάρτηση επίδοσης από τους compilers και τα compiler flags.
 - Χρήση βιβλιοθηκών και εξάρτηση επίδοσης από αυτές.
 - Χρήση εργαλείων υψηλού επιπέδου γλώσσας : Octave.

Πολλαπλασιασμός Πινάκων : Ορισμοί

$$A \cdot B = C$$

$$\begin{bmatrix} A_{11} & A_{12} & \dots & A_{1N} \\ A_{21} & A_{22} & \dots & A_{2N} \\ \dots & \dots & \dots & \dots \\ A_{M1} & A_{M2} & \dots & A_{MN} \end{bmatrix} \times \begin{bmatrix} B_{11} & B_{12} & \dots & B_{1K} \\ B_{21} & B_{22} & \dots & B_{2K} \\ \dots & \dots & \dots & \dots \\ B_{N1} & B_{N2} & \dots & B_{NK} \end{bmatrix} = \begin{bmatrix} C_{11} & C_{12} & \dots & C_{1K} \\ C_{21} & C_{22} & \dots & C_{2K} \\ \dots & \dots & \dots & \dots \\ C_{M1} & C_{M2} & \dots & C_{MK} \end{bmatrix}$$

- Για κάθε στοιχείο του πίνακα C απαιτούνται N πολλαπλασιασμοί και N προσθέσεις.
- Συνολικά $N \times M \times K$ πολλαπλασιασμοί και $N \times M \times K$ προσθέσεις, $= 2 \times N \times M \times K$ πράξεις κινητής υποδιαστολής.
- Για $M = N = K = 1000$, χρειάζονται 2×10^9 πράξεις. Εάν μια μηχανή κάνει αυτές τις πράξεις σε 1 δευτερόλεπτο, τότε η απόδοσή της είναι 2 GFlops.

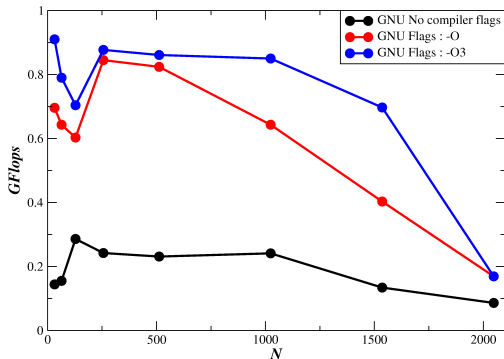
Πολλαπλασιασμός Πινάκων

- Τα nodes του ARIS έχουν επίδοση 422 GFlops= 21 GFlops/core.
- Το ARIS έχει συνολική επίδοση 180 TFlops.
- Να το δούμε στην πράξη : Κώδικας σε Fortran, $N=M=K$

```
do i = 1, N
  do j = 1, N
    c(i,j) = 0.0000000000000000
    do ij = 1, N
      c(i,j) = c(i,j) + a(i,ij) * b(ij,j)
    enddo
  enddo
enddo
```

Πολλαπλασιασμός Πινάκων

- GNU Compilers



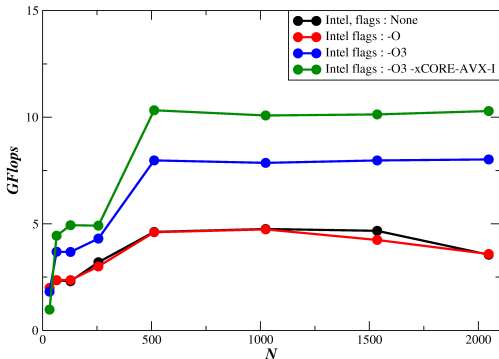
Πολλαπλασιασμός Πινάκων

- Παρατηρήσεις :

- Σημαντική διαφορά απόδοσης αναλόγως compiler flags
- Σημαντική διαφορά απόδοσης αναλόγως μεγέθους (N)
- Αυξάνοντας το N η απόδοση πέφτει για $N > 1000$
- Πολύ μικρότερη απόδοση (25-250 φορές μικρότερη) από την θεωρητική/μετρημένη
- Γιατί ?
- Compiler, flags, L1/L2 Cache, Instruction Set,

Πολλαπλασιασμός Πινάκων

- Intel Compilers

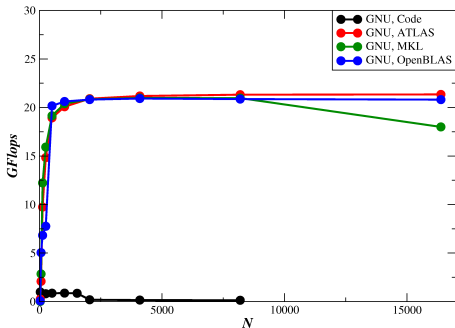


Πολλαπλασιασμός Πινάκων

- Παρατηρήσεις :
 - Σημαντική βελτίωση απόδοσης σε όλα τα N και flags
 - Σημαντική διαφορά απόδοσης αναλόγως compiler flags
 - Διαφορά απόδοσης αναλόγως μεγέθους (N), αλλά λιγότερο σημαντική
 - Μικρότερη απόδοση (2-5 φορές μικρότερη) από την θεωρητική/μετρημένη

Πολλαπλασιασμός Πινάκων

- GNU Compilers, Optimized Libraries
- Αντικατάσταση του κώδικα που είδαμε πιο πριν με :
`call dgemm('N','N',N,N,N,ONE,a,lda,b,ldb,ONE,c,ldc)`



Πολλαπλασιασμός Πινάκων

- Παρατηρήσεις :
 - Σημαντική βελτίωση απόδοσης σε όλα τα N.
 - Μεγάλη διαφοροποίηση απόδοσης ως προς απλό κώδικα.
 - Σταθεροποίηση απόδοσης ως προς μέγεθος.
 - Απόδοση χρησιμοποιώντας τις optimized βιβλιοθήκες πολύ κοντά στην θεωρητική/μετρημένη απόδοση.
 - Σχετικά μικρές διαφοροποιήσεις απόδοσης αναλόγως βιβλιοθήκης.

Γενική Χρήση του ARIS

Δρ. Δημήτρης Ντελλής

Περιεχόμενα

Environment Modules

Χρήση των modules

Batch System

SLURM

Εντολές SLURM

SLURM Limits

PBS emulation

Accounting

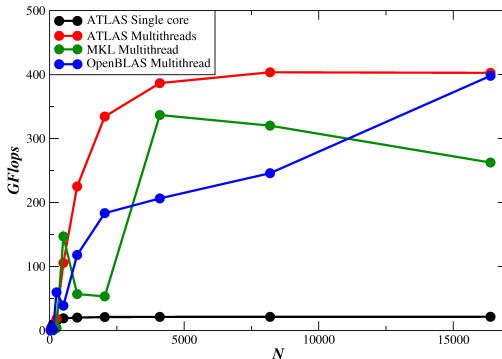
Διαθέσιμα πακέτα

Βέλτιστες Πρακτικές - Κοινά προβλήματα

Πρακτική εξάσκηση

Πολλαπλασιασμός Πινάκων

- GNU Compilers, SMP Optimized Libraries



Πολλαπλασιασμός Πινάκων

- Παρατηρήσεις :
 - Σημαντική αύξηση επίδοσης, περίπου ανάλογη με τον αριθμό cores, σε όλα τα N.
 - Τεράστια (500x) διαφοροποίηση επίδοσης ως προς απλό, single core κώδικα.
 - Σημαντικές διαφοροποιήσεις επίδοσης αναλόγως N.
 - Η ATLAS συστηματικά πιο γρήγορη σε όλα τα N μέχρι 16384, με συγκλίνουσα επίδοση.
 - Η OpenBLAS αν και σε μικρά N υπολείπεται της ATLAS, σε μεγάλα N είναι αρκετά καλή με αυξητική τάση επίδοσης.
 - Η MKL παρουσιάζει αρκετές διακυμάνσεις επίδοσης σε διάφορα N, με πτωτική τάση σε μεγάλα N.
 - Σε μεγάλα N, απόδοση πολύ κοντά στην θεωρητική/μετρημένη επίδοση των nodes.

Πολλαπλασιασμός Πινάκων

- Τελειώσαμε εδώ φτάνοντας πολύ κοντά στη θεωρητική επίδοση σε μεγάλα N ?
- Σε κάποιες περιπτώσεις εφαρμογών, γίνεται μεγάλος αριθμός πολλαπλασιασμών πινάκων μικρών διαστάσεων με διαστάσεις που είναι σε μεγάλο ποσοστό πρώτοι αριθμοί, π.χ. $A(11,7) \times B(3,11)$. Οι optimized BLAS συνήθως δεν έχουν ικανοποιητική επίδοση.
- **LibSMM** (Small Matrices Multiplication) developed by Cray Inc.
 - Επιλέγοντας compiler, flags και κάποια BLAS αναφοράς, βρίσκει την επίδοση για $N/M/K$ στο εύρος 1-32, και αφού κάνει αρκετές χρονομετρήσεις αποφασίζει για κάθε συνδυασμό $N/M/K$ αν θα χρησιμοποιήσει το δικό της κώδικα ή την βιβλιοθήκη αναφοράς.

Πολλαπλασιασμός Πινάκων

- Αντικατάσταση είτε του κώδικα είτε της κλήσης dgemm με **call smm_dnn(M,N,K,A,B,C)**.
- Χρονομετρώντας 2000 πολλαπλασιασμούς πινάκων όλων των διαστάσεων $N, M, K=1-32$, η μέση επίδοση σε GFlops είναι :

Code	ATLAS	LibSMM
1.63	5.02	5.21

- Μέγιστη σχετική επίδοση LibSMM ως προς ATLAS για τα ίδια μεγέθη : 10.95.

Πολλαπλασιασμός Πινάκων

- Ακολουθώντας πρακτικές βέλτιστης χρήσης, επιταχύνθηκε ο πολλαπλασιασμός πινάκων έως ~ 3000 φορές σε ένα μόνο node : ATLAS SMP vs code, gnu, no flags.

Χρήση εργαλείων υψηλού επιπέδου γλώσσας : Octave

- Τί είναι : GNU Octave is a high-level interpreted language, primarily intended for numerical computations. It provides capabilities for the numerical solution of linear and nonlinear problems, and for performing other numerical experiments. It also provides extensive graphics capabilities for data visualization and manipulation. Octave is normally used through its interactive command line interface, but it can also be used to write non-interactive programs. The Octave language is quite similar to Matlab so that most programs are easily portable.

Χρήση εργαλείων υψηλού επιπέδου γλώσσας : Octave

- Παράδειγμα χρήσης Octave

```
cat matmul.m
```

```
a=double(rand(16384,16384));
```

```
b=double(rand(16384,16384));
```

```
tzero=time();
```

```
c=a*b;
```

```
tnow=time()-tzero
```

- Κάνει πολλαπλασιασμό πινάκων διαστάσεων 16kx16k.
- Το εγκατεστημένο octave στο ARIS χρησιμοποιεί optimized βιβλιοθήκες για οτιδήποτε υπάρχει διαθέσιμο σε optimized version.

Γενική Χρήση
του ARIS

Δρ. Δημήτρης
Ντελλής

Περιεχόμενα

Environment
Modules

Χρήση των modules

Batch System

SLURM

Εντολές SLURM

SLURM Limits

PBS emulation

Accounting

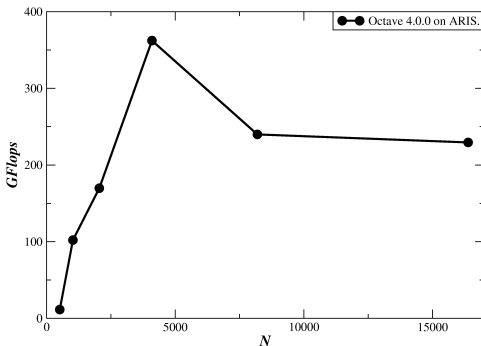
Διαθέσιμα
πακέτα

Βέλτιστες
Πρακτικές -
Κοινά
προβλήματα

Πρακτική
εξάσκηση

Χρήση εργαλείων υψηλού επιπέδου γλώσσας : Octave

- Octave Performance on ARIS single Node, 20 threads/task



Πολλαπλασιασμός Πινάκων

- Μέγεθος προβλήματος :
- Δεδομένου ότι τα nodes έχουν διαθέσιμη 56 GB RAM, μπορούμε να κάνουμε πολλαπλασιασμό πινάκων με ίδιες διαστάσεις, υποθέτοντας ότι δεν έχουμε άλλες μεταβλητές που χρειάζονται πολύ μνήμη, μέχρι ~ 50000.
- Για μεγαλύτερα μεγέθη : Κατανομή των στοιχείων των πινάκων στα παράλληλα process
 - BLACS/Scalpack : Συνδιασμός Partitioner/Solver. Χρήση παρόμοια με αυτή του LAPACK, αλλά με περισσότερες λεπτομέρειες/παραμέτρους.

Πέραν του Πολλαπλασιασμού Πινάκων

- Στο ARIS υπάρχουν ήδη εγκατεστημένες/βελτιστοποιημένες διάφορες βιβλιοθήκες που επιλύουν συγκεκριμένα μαθηματικά προβλήματα. Ενδεικτικά αναφέρεται ένας κατάλογος μεθόδων μιας μόνο βιβλιοθήκης, της GSL:
http://www.gnu.org/software/gsl/manual/html_node/
- Ο κατάλογος των βιβλιοθηκών/εφαρμογών εμπλουτίζεται συνεχώς.