ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

“ARIS SUPERCOMPUTER”

Γεώργιος Ι. Ναυπακτίτης

Λεωνίδας Φ. Δημάκης

Επιβλέπων: **Κοτρώνης Ιωάννης,** Καθηγητής

ΑΘΗΝΑ

ΙΟΥΝΙΟΣ 2016

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

“ARIS SUPERCOMPUTER”

Γεώργιος Ι. Ναυπακτίτης
Α.Μ.: 1115200900238

Λεωνίδας Φ. Δημάκης
Α.Μ.: 1115200600243

ΕΠΙΒΛΕΠΟΝΤΕΣ: **Κοτρώνης Ιωάννης,** Καθηγητής

ΠΕΡΙΛΗΨΗ

Στη παρούσα πτυχιακή εργασία καταγράφουμε τις πληροφορίες και τα συμπεράσματα που αντλήσαμε από την ενασχόληση μας με τον Supercomputer A.R.I.S. που εγκαταστάθηκε και λειτουργεί στο υπουργείο παιδείας για να συμβάλει στην επιστημονική έρευνα. Στόχος της πτυχιακής μας είναι να αποτελέσει έναν οδηγό για τους μελλοντικούς χρήστες αυτού του υπολογιστικού συστήματος. Οι χρήστες του A.R.I.S. δεν προέρχονται μόνο από το κλάδο της πληροφορικής, γι' αυτό το λόγο προσπαθήσαμε να απλουστεύσουμε τα βήματα και εμπλουτίσαμε το κείμενο με περισσότερες πληροφορίες για κάποιες εξειδικευμένες έννοιες που θα συναντήσουμε, έτσι ώστε να γίνουν κατανοητές σε όλους.

Στο πρώτο κεφάλαιο κάνουμε μια γενική επισκόπηση του υπολογιστικού συστήματος A.R.I.S. με τα τεχνικά του χαρακτηριστικά και τον αποθηκευτικό του χώρο. Στη συνέχεια στο δεύτερο κεφάλαιο δίνουμε οδηγίες χρήσης του συστήματος και οδηγίες για εκτέλεση προσωπικών προγραμμάτων ή εφαρμογών που υπάρχουν εγκατεστημένα στο σύστημα. Στο τελευταίο κεφάλαιο παραθέτουμε ένα δικό μας project που τρέξαμε στον A.R.I.S. και παρουσιάζουμε τρόπους βελτίωσης παράλληλων προγραμμάτων με τη βοήθεια εφαρμογών που υποστηρίζει το σύστημα.

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ: Παράλληλος Προγραμματισμός,Υπερυπολογιστές

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: Mpi,parallel systems,parallel computing,supercomputers,SLURM,mpiP

ΠΕΡΙΕΧΟΜΕΝΑ

ΚΑΤΑΛΟΓΟΣ ΣΧΗΜΑΤΩΝ

ΚΑΤΑΛΟΓΟΣ ΕΙΚΟΝΩΝ

ΚΑΤΑΛΟΓΟΣ ΠΙΝΑΚΩΝ

ΠΡΟΛΟΓΟΣ

Η πτυχιακή αυτή εργασία αποτελεί την κορύφωση των σπουδών μας στο Εθνικό και Καποδιστριακό Πανεπιστήμιο Αθηνών, Τμήμα Πληροφορικής Και Τηλεπικοινωνιών. Δεδομένου ότι ήταν εργασία δύο ατόμων οφείλουμε να πούμε ότι ο φόρτος μοιράστηκε ισάξια και στα δυο μέλη και κατά τη διάρκεια της διεξαγωγής η συνεργασία ήταν άψογη. Η πρόσβαση στον supercomputer A.R.I.S. παρέχονταν αποκλειστικά μέσω του δικτύου του πανεπιστημίου μας και του υπερυπολογιστή που έχει εγκατασταθεί στο κτήριο του τμήματος Πληροφορικής και Τηλεπικοινωνιών.

Στο σημείο αυτό θα θέλαμε να ευχαριστήσουμε τον επιβλέποντα καθηγητή μας Κ.Ιωάννη Κοτρώνη για την στήριξη του και για την ευκαιρία που μας έδωσε να ασχοληθούμε σε βάθος με το παράλληλο προγραμματισμό, εξασφαλίζοντας τη πρόσβαση μας σε αυτό το τόσο υψηλού επιπέδου υπολογιστικό σύστημα. Ευχαριστούμε επίσης το προσωπικό του A.R.I.S. για την υποστήριξη που μας παρείχε μέσω e-mail κατά τη διάρκεια της πρόσβασης μας σ' αυτόν.

1.ΕΠΙΣΚΟΠΗΣΗ ΤΟΥ SUPERCOMPUTER A.R.I.S.

ARIS είναι το όνομα του νέου υπερυπολογιστή, που έχει εγκατασταθεί και λειτουργεί από το ΕΔΕΤ (Ελληνικό Δίκτυο Έρευνας και Τεχνολογίας) στην Αθήνα. Το όνομα ARIS σημαίνει Advanced Research Information System. Πρόκειται για ένα σύστημα με στόχο να στηρίξει προγράμματα ερευνητικού σκοπού γραμμένα για συστήματα παράλληλης αρχιτεκτονικής.
Το σύστημα έχει μια θεωρητική μέγιστη απόδοση (Rpeak) των 190,85 TFLOPS και μια σταθερή απόδοση (Rmax) των 179,73 TFLOPS στο Benchmark* Linpack. Το σύστημα κατετάγη #468 στη λίστα των TOP500 πιο ισχυρών συστημάτων στον κόσμο, όταν εγκαταστάθηκε (Ιούνιος 2015).

*Τα Benchmarks είναι προγράμματα αξιολόγησης παράλληλων συστημάτων όπως ο ARIS.

1.1. Τεχνικά Χαρακτηριστικα

ARIS compute nodes	
Architecture	x86-64
System	IBM NeXtScale nx360 M4
Total number of nodes	426
Total number of cores	8520
Total amount of RAM [TByte]	27
Total Linpack Performance [TFlop/s]	180
Components	
Processor Type	Ivy Bridge - Intel Xeon E5-2680v2
Nominal Frequency [GHz]	2.8
Processors per Node	2
Cores per Processor	10
Cores per Node	20
Hyperthreading	OFF
Memory	
Memory per Node [GByte]	64 (usable 57)
Interconnect	
Technology	Infiniband FDR
Topology	Fat tree
Bandwidth [Gb/s]	56
Storage	
Type	IBM GPFS
Size [PByte]	1

Bandwidth [GB/s]	6
System Software	
Operating system	RedHat Linux 6.6
Batch system	SLURM
System Management	xCat IBM
Monitoring	Nagios, Ganglia

Login nodes	
Number of Nodes	2
Processor Type	Intel(R) Xeon(R) CPU E5-2640 v2
Nominal Frequency [GHz]	2.00GHz
Processors per Node	2
Cores per Processor	16
Threads per Processor	32
Hyperthreading	ON
Memory per Node [GByte]	128

Εικόνα 1.1: Τεχνικά χαρακτηριστικά του A.R.I.S.

Επεξήγηση τεχνικών χαρακτηριστικών

- Ο ARIS αποτελείτε από δύο ήδη nodes. Ο ένας τύπος είναι για το computing(backend/ computing nodes) δηλαδή την εκτέλεση των προγραμμάτων.Οι συνολικοί nodes αυτού του τύπου είναι 426. Ο άλλος τύπος (frontend/login nodes) είναι για την εξυπηρέτηση των χρηστών και έχει ως αρμοδιότητα την δημιουργία και διαχείριση ουράς των jobs/ programs που έχουν αιτήσει οι χρήστες τα οποία περιμένουν με σειρά προτεραιότητας για να εκτελεστούν από τους backend nodes. Ο χρόνος αναμονής εξαρτάται από την διαθεσιμότητα των πόρων του backend συστήματος. Ο αριθμός αυτών το nodes στον ARIS είναι 2.

- **System:** IBM NeXtScale nx360 M4: Ο τύπος του υπολογιστικού συστήματος κάθε node.

- **Type:** IBM GPFS: Ο τύπος του file system που χρησιμοποιεί ο ARIS (**General Parallel File System)**

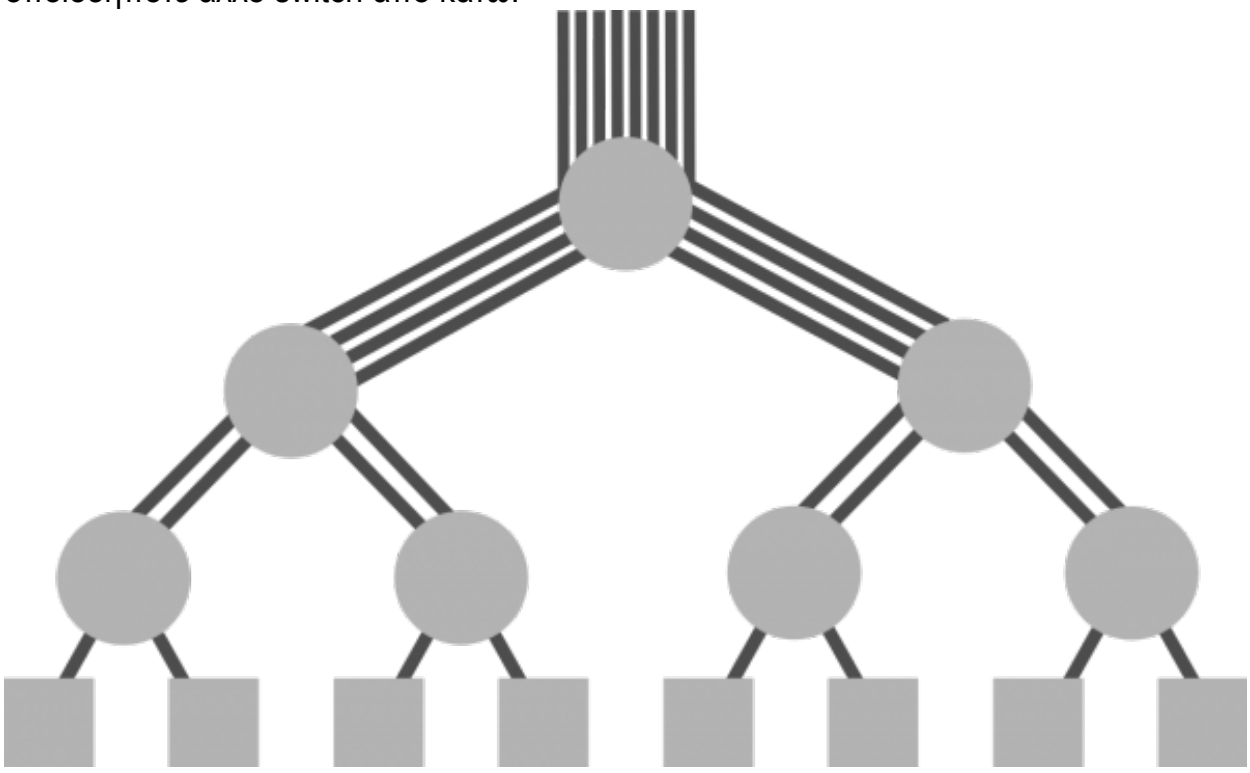
- **Operating system:** RedHat Linux 6.6: Το 97% των supercomputers παγκοσμίως χρησιμοποιεί λειτουργικό σύστημα linux

- **Hyperthreading:** ON/OFF :

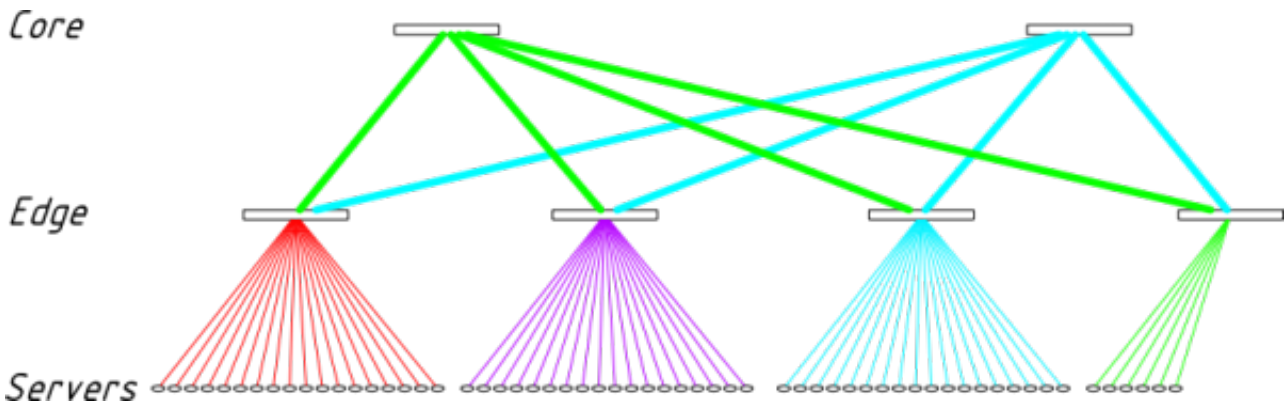
Η τεχνολογία αυτή της Intel, επιτρέπει την παράλληλη επεξεργασία δεδομένων (πολλές διεργασίες/νήματα ταυτόχρονα) στους επεξεργαστές που κατασκευάζει. Κάθε επεξεργαστικός πυρήνας, γίνεται αντιληπτός από το λειτουργικό σύστημα ως δυο λογικοί/ φανταστικοί πυρήνες (συγκεκριμένα ένας πραγματικός επεξεργαστικός πυρήνας και ένας επιπλέον "λογικός"), και διαμοιράζει το φόρτο ανάμεσά τους όταν παρίσταται η ανάγκη. Η ουσιαστική λειτουργία της υπερνημάτωσης είναι η μείωση του αριθμού των εξαρτημένων εντολών στον δίαυλο μεταφοράς εντολών του επεξεργαστή.
Η υπερνημάτωση απαιτεί την ύπαρξη λειτουργικού συστήματος το οποίο όχι μόνο θα πρέπει να υποστηρίζει πολλαπλούς πυρήνες επεξεργασίας αλλά και να είναι ειδικά σχεδιασμένο να εκμεταλλεύεται τη τεχνολογία αυτή.

- **Topology:** Fat tree :

Σε ένα switched fabric - μια τοπολογία δικτύου που χρησιμοποιεί switches - ο κύριος στόχος είναι να συνδεθεί ένας μεγάλος αριθμός τερματικών (επεξεργαστές ή servers), χρησιμοποιώντας switches που έχουν μόνο έναν περιορισμένο αριθμό ports.Με μια έξυπνη τοπολογία συνδεδεμένων switches μπορεί να διασυνδεθεί ένα εντυπωσιακό ποσό τερματικών.
Η τοπολογία Fat-Tree προτάθηκε από τον Charles E. Leiserson το 1985. Η εν λόγο τοπολογία είναι ένα δέντρο, και τα τερματικά(servers) συνδέονται στα φύλλα του δέντρου. Το ιδιαίτερο χαρακτηριστικό του fat-tree είναι ότι για κάθε switch, ο αριθμός των συνδέσεων με τα switches του από κάτω επιπέδου είναι ίσος με τον αριθμό των συνδέσεων των switches του από πάνω επιπέδου. Ως εκ τούτου, οι συνδέσεις των switches που βρίσκονται σε μεγαλύτερα επίπεδα είναι όλο και περισσότερες, και το switch που βρίσκεται στη ρίζα του δέντρου έχει τις περισσότερες συνδέσεις σε σχέση με οποιοδήποτε άλλο switch από κάτω:



Εικόνα 1.2: Fat Tree topology I



Εικόνα 1.3: Fat Tree topology II

Πόσα τερματικά θα μπορούσαν να συνδεθούν σε ένα δίκτυο με fat tree τοπολογία;
Έστω ότι τα edge switches έχουν Pe ports, και τα core switches έχουν Pc ports, τότε ένα Fat tree δίκτυο δύο επιπέδων θα μπορεί να συνδέσει το πολύ μέχρι Nmax = Pe * PC / 2 τερματικά. Ας υποθέσουμε ότι χρησιμοποιούμε ένα από τα μεγαλύτερα switches που έχουν κατασκευαστεί το οποίο διαθέτει P = 648 ports, και χρησιμοποιούμε το ίδιο μοντέλο switch και στα δύο επίπεδα edge και core (Pe = Pc = P), τότε Nmax = P * P / 2 = 648 * 648 / 2 = 209,952 τερματικά. Αυτός ο αριθμός τερματικών είναι αρκετός για πολλούς supercomputers! Αν χρειαστεί να συνδέσουμε ακόμη περισσότερα τερματικά, μια τοπολογία τριών επιπέδων θα ήταν αρκετή.

- **Technology:** Infiniband FDR :

Ο ARIS χρησιμοποιεί Infiniband switches για τη fat-tree τοπολογία.Το InfiniBand (συντομογραφία IB), είναι ένα πρότυπο επικοινωνίας τερματικών σαν το ethernet που χρησιμοποιείτε σε high performance computers. Διαθέτει πολύ υψηλή απόδοση και πολύ χαμηλό latency. Από το 2014 είναι ο πιο συχνά χρησιμοποιούμενος τρόπος διασύνδεσης επεξεργαστικών μονάδων σε supercomputers. Το πρότυπο αυτό έχει αρχίσει να αντικαθιστά το PCI bus σε high-end servers και PCs.

- **Batch System:** SLURM :

Batch processing είναι η αυτόματη εκτέλεση μιας σειράς προγραμμάτων («jobs») σε έναν υπολογιστή.Τα jobs έχουν συσταθεί να εκτελεστούν χωρίς την ανθρώπινη αλληλεπίδραση. Όλες οι παράμετροι εισόδου(τα arguments του εκτελέσιμου) είναι προκαθορισμένες μέσω scripts. Αυτό σημαίνει ότι τα προγράμματα δεν έχουν αλληλεπίδραση με το χρήστη κατά τη διάρκεια της εκτέλεσης τους, για παράδειγμα δεν μπορούν να πάρουν input από το command prompt όταν τρέχουν. Ένα πρόγραμμα παίρνει σαν είσοδο ένα σύνολο αρχείων με δεδομένα, επεξεργάζεται τα δεδομένα, και παράγει μια σειρά από αρχεία εξόδου με τα αποτελέσματα.

Slurm Workload Manager(Batch System)

Το Slurm είναι ένα open-source πρόγραμμα διαχειρισμού του φόρτου εργασίας του backend ενός supercomputer σχεδιασμένο για λειτουργικό Linux. Παρέχει τρεις βασικές λειτουργίες. Πρώτον, διαθέτει αποκλειστική ή / και μη-αποκλειστική πρόσβαση σε πόρους (computer nodes ενός supercomputer), για τους χρήστες για κάποιο χρονικό διάστημα ώστε να μπορούν να εξυπηρετηθούν εκτελώντας ένα σύνολο από jobs. Δεύτερον, παρέχει ένα framework για την έναρξη, εκτέλεση και έλεγχο/παρακολούθηση των jobs σε ένα σύνολο κατοχυρωμένων nodes απ το χρήστη. Τέλος, διαχειρίζεται την ουρά των εκκρεμών jobs.

Περισσότερες πληροφορίες σχετικά με το SLURM υπάρχουν παρακάτω στις οδηγίες χρήστη και στο Documentation του ARIS(http://doc.aris.grnet.gr/run/)

- **System Management:** xCat IBM:

Το Extreme Cloud Administration Toolkit (xCAT) είναι ένα open source εργαλείο που παρέχει ένα ενιαίο interface για hardware control, discovery, και OS diskfull / diskfree ανάπτυξη. Αυτό το toolkit χρησιμοποιείται για την ανάπτυξη και διαχείριση Linux clusters.Το xCAT κάνει και απλή και εύκολη την:

-Εγκατάσταση Linux cluster με βοηθητικά προγράμματα για την εγκατάσταση πολλών μηχανών σε παράλληλη αρχιτεκτονική.

-Διαχείριση του Linux cluster με εργαλεία για τη διαχείριση παράλληλων λειτουργιών.

-Εγκατάσταση προγραμμάτων* για high performance computing, συμπεριλαμβανομένου του λογισμικού batch(όπως το Slurm για παράδειγμα), parallel libraries, και άλλο λογισμικό που είναι χρήσιμο για το cluster.

*Τα εγκατεστημένα προγράμματα του ARIS για high performance computing βρίσκονται στο παράρτημα Software Invironment στη σελίδα http://doc.aris.grnet.gr/

- **Monitoring:** Nagios, Ganglia: Λογισμικό για έλεγχο/παρακολούθηση του συστήματος ARIS το οποίο δίνει πληροφορίες σχετικά με τη κατάσταση των διάφορων parts του.

1.2. Επισκόπηση Αποθηκευτικού Χώρου

Υπάρχουν δύο κοινά file systems για τον ARIS, το HOME και το WORKDIR. Όλα τα login και compute nodes έχουν πρόσβαση στα ίδια δεδομένα στα κοινά file systems.

File system	Type	Env. Variable	Size	Lifetime	Backup	Quota	Description
/users	GPFS	$HOME	250 TB	Project duration	NO	*	source files, input data
/work	GPFS	$WORKDIR	500 TB	Project duration	NO	*	short term, fast storage

Εικόνα 1.4: A.R.I.S. File systems

$HOME

Οι χρήστες ΔΕΝ θα πρέπει να τρέχουν τα jobs από αυτό το file system, λόγω των χαμηλών του επιδόσεων. Αυτό το file system δίνει έμφαση στην αξιοπιστία πάρα στις επιδόσεις.

Αποθηκεύστε τον πηγαίο κώδικα σας και κάντε build τα εκτελέσιμα σας εδώ.
Η μεταβλητή περιβάλλοντος $HOME δείχνει στο προσωπικό φάκελο του χρήστη /users/[username].

$WORKDIR

Το /work file system δίνει έμφαση στην απόδοση παρά την αξιοπιστία, ως ένας γρήγορος χώρος εργασίας για προσωρινή αποθήκευση.

Αποθηκεύστε μεγάλα αρχεία εδώ.

Αλλάξτε σε αυτόν τον κατάλογο στο batch και τρέξτε τα jobs σ' αυτό το file system. Η μεταβλητή περιβάλλοντος $WORKDIR δείχνει στο προσωπικό φάκελο του χρήστη /users/[username].

Quota

Disk quota management είναι τα δικαιώματα που δίνονται από τους διαχειριστές θέτοντας όρια στο χρήστη, σε ομάδες χρηστών, ή άλλες ομάδες του χώρου αποθήκευσης. Με τον καθορισμό quota, προλαμβάνεται ο κίνδυνος να μένει εκτός αποθηκευτικού χώρου το σύστημα. Έτσι οι πόροι μοιράζονται στους χρήστες μέσω quotas αναλόγως τις ανάγκες του project για το οποίο έχουν αιτήσει χρήση του ARIS.

*** Σε κάθε περίπτωση, εάν το μέγεθος του αποθηκευτικού χώρου που χορηγείται είναι ανεπαρκές, μπορείτε να επικοινωνήστε με την υποστήριξη, τα quota μπορούν να αυξηθούν κατόπιν αιτήματος.**

Quota commands

Τα quota των χρηστών στο κάθε file system μπορούν να ελεγχθούν με mmlsquota:

```
$ mmlsquota --block-size 1G
                         Block Limits                                    |     File Limits
Filesystem type             GB      quota      limit   in_doubt    grace |    files   quota    limit in_d
gpfs0      USR              14       1000       1200          0     none |     1008       0        0

                         Block Limits                                    |     File Limits
Filesystem type             GB      quota      limit   in_doubt    grace |    files   quota    limit in_d
users      USR              27       1000       1200          0     none |   110914       0        0
```

Εικόνα 1.5: mmlsquota command

Για κάθε file system στο cluster η εντολή **mmlsquota** μας δείχνει τα παρακάτω:

1.Block limits:

quota type (USR or GRP or FILESET)
current usage
soft limit
hard limit
space in doubt
grace period

2.File limits:

current number of files
soft limit
hard limit
files in doubt
grace period

3.Note**:**

In cases where small files do not have an additional block allocated for them, quota usage may show less space usage than expected.

4**.**Remarks

Ελέγξτε το disk usage με την εντολή du:
Παράδειγμα: $HOME directory

```
$ du -chs $HOME
```

Πολιτική File system(Σημαντικές Πληροφορίες):

Δεν υπάρχει BACKUP για οποιαδήποτε αρχεία των χρηστών ούτε στο $HOME ούτε $WORKDIR file system.

Δεν υπάρχει κανένας τρόπος για να ανακτηθούν διαγραμμένα δεδομένα.
Οι χρήστες πρέπει να κρατούν backup των δεδομένων σε τοπικό δίσκο τους.

2.ΟΔΗΓΙΕΣ ΧΡΗΣΗΣ ΤΟΥ SUPERCOMPUTER A.R.I.S.

2.1. Αίτηση και προϋποθέσεις για αποδοχή της αίτησης

Για να αποκτήσετε πρόσβαση και να χρησιμοποιήσετε τον ARIS θα πρέπει να κάνετε αίτηση. Περισσότερα για της αιτήσεις και τη πολιτική αποδοχής τους επισκευθείτε τον παρακάτω σύνδεσμο.
https://hpc.grnet.gr/access_policy/

2.2. Login και Data transfer Αφού εγκριθεί η αίτηση

- Για να κάνουμε login θα πρέπει να έχουμε δημιουργήσει πρώτα ένα συνδυασμό κλειδιών ssh* και να έχουμε στείλει το public key στους διαχειριστές του ARIS. Στη συνέχεια εκτελούμε στο terminal του προσωπικού μας υπολογιστή(ο οποίος έχει την ip που έχουμε δώσει επίσης στους διαχειριστές του ARIS) την εντολή και στη συνέχεια πληκτρολογείτε το προσωπικό password/passphrase που έχετε δημιουργήσει για το private key σας:
  ssh -i /path/to/rsa(private key,most common: ~/.ssh/rsa) -YC username@login.aris.grnet.gr

- Για μεταφορά αρχείων το σύστημα χρησιμοποιεί sftp. Εάν δεν έχετε κάποιο αντίστοιχο πρόγραμμα για sftp χρησιμοποιείστε τις παρακάτω εντολές:

  1. Login : sftp -oIdentityFile=/path/to/rsa username@login.aris.grnet.gr

  2. Transfer file: put local/file/path remote/file/path
     get remote/file/path local/file/path

  3. Transfer dir:

     α)Για να μεταφέρουμε φακελο πρεπει να τον συμπιέσουμε με την εντολή:
     tar -czvf archivename.tar.gz directory
     β)Και μετά τον μεταφέρουμε με τις εντολές όπως στο Transfer file

  4. Για αποσυμπίεση: tar -zxvf archivename.tar.gz

*Εάν δεν γνωρίζετε τι είναι τα ssh keys διαβάστε το παρακάτω κεφάλαιο.

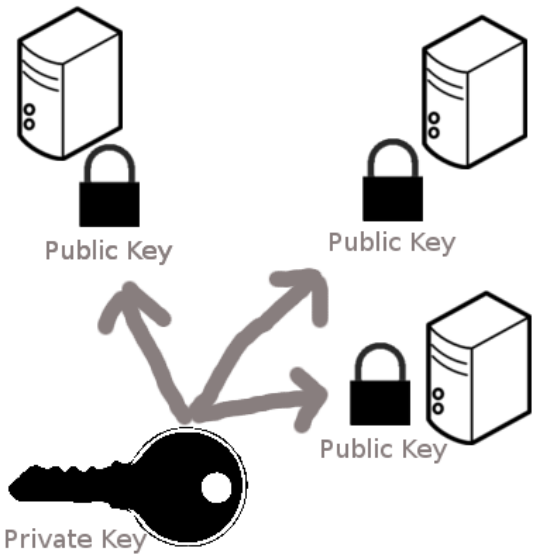
2.3. Τι είναι τα ssh public και private keys

•Σκεφτείτε ένα public κλειδί σαν μια κλειδαριά. Δεν είναι πραγματικά ένα κλειδί, είναι ένα λουκέτο που μπορείτε να κάνετε πολλά αντίγραφα και να τα διανείμετε όπου και σε όποιον θέλετε. Για παράδειγμα, αν θέλετε να βάλετε «λουκέτο» σας σε λογαριασμό SSH σε ένα άλλο μηχάνημα, θα το αντιγράψετε στα «authorized_keys» στο ~ / .ssh φάκελο.

•Σκεφτείτε ένα private κλειδί, όπως είναι το πραγματικό κλειδί. Αυτό είναι που χρησιμοποιήσετε για να ανοίξετε το λουκέτα που είναι αποθηκευμένα σε άλλα μηχανήματα. Ακριβώς όπως ένα κανονικό κλειδί θα πρέπει να το κρατάτε ασφαλές μέρος, και μακριά από λάθος χέρια.

Εικόνα 2.1: Private/Public Keys I

Το public κλειδί σας (λουκέτο) μπορεί να διανεμηθεί παντού. Όσο το private κλειδί σας δεν είναι εκτεθειμένο, δεν υπάρχει κανένας κίνδυνος. Ακριβώς όπως στην αναλογία λουκέτου/ κλειδιού, δεν θα σας ένοιαζε αν υπήρχαν εκατοντάδες ίδια λουκέτα , εφ 'όσον είστε ο μόνος που έχετε το κλειδί.
Έτσι, όταν τρέξετε την εντολή ‘ssh-keygen' στο terminal θα παραχθούν το private (id_rsa) και το public (id_rsa.pub) κλειδί προς χρήση. Οπότε έχετε την κλειδαριά και το κλειδί για να την ανοίξετε. Μπορείτε να δημιουργήσετε αντίγραφα του id_rsa.pub (δημόσιο κλειδί / λουκέτο) και να τα αποθηκεύσετε σε υπολογιστές που θέλετε κάποιος να συνδεθεί remote με την εντολή ssh χρησιμοποιώντας το private κλειδί.



Εικόνα 2.2: Private/Public keys II

Τώρα, αν θέλετε να πάτε ένα βήμα παραπέρα ssh-keygen σας επιτρέπει να βάλετε έναν κωδικό πρόσβασης για το private κλειδί(ο κωδικός αυτός θα σας ζητείτε κάθε φόρα που επιχειρείτε remote login με ssh). Αυτό προσθέτει ένα ακόμη στρώμα προστασίας. Ακόμη και αν το private κλειδί σας πέσει σε λάθος χέρια, θα πρέπει να γνωρίζουν τον κωδικό πρόσβασης για να το χρησιμοποιήσουν.

Για δημιουργία συνδυασμού ssh κλειδιών εκτελέστε τις παρακάτω εντολές στο terminal:
mkdir ~/.ssh
chmod 700 ~/.ssh
ssh-keygen -t [private_key]
chmod 600 /path/to/[private_key]

Το public key σας είναι διαθέσιμο στο ~/.ssh/[public_key]
Το private key σας είναι διαθέσιμο στο ~/.ssh/[private_key]

2.4. Environment και Modules

Environment Variables

Τα Environment Variables είναι ένα σύνολο μεταβλητών που περιέχουν δεδομένα τα οποία χρησιμοποιούνται από μία ή περισσότερες εφαρμογές. Είναι μεταβλητές με ένα όνομα και μια τιμή. Η τιμή ενός environment variable μπορεί για παράδειγμα να είναι το path όλων των εκτελέσιμων στο file system, ο default editor που πρέπει να χρησιμοποιείται , ή οι τοπικές ρυθμίσεις του συστήματος.Για να δείτε τα environment variables χρησιμοποιήστε την εντολή env.

Environment Modules

Ο ARIS χρησιμοποιεί το environment module για να διαχειρίζεται στο περιβάλλον του χρήστη τα διάφορα προγράμματα, libraries και compiler versions που έχει. Το πακέτο Environment Modules παρέχει τη δυναμική τροποποίηση του περιβάλλοντος του χρήστη μέσω module αρχείων. Το σαφές πλεονέκτημα των modules είναι ότι διαχειρίζονται τα environment variables όπως τα PATH, LD_LIBRARY_PATH κλπ και οι χρήστες απλά κάνουν load και unload τα modules για να ρυθμίσουν το περιβάλλον τους.

Module Usage

Η εντολή module χρησιμοποιεί υπο-εντολές για να εκτελέσει διάφορες ενέργειες.

Για να εμφανιστούν όλα τα διαθέσιμα modules εκτελέστε: module avail

```
module avail

------------------------- /apps/modulefiles/compilers -------------------------
binutils/2.25      gnu/4.9.3          gnu/5.2.0          pgi/15.5
gnu/4.9.2(default) gnu/5.1.0          intel/15.0.3

------------------------- /apps/modulefiles/parallel --------------------------
intelmpi/5.0.3      openmpi/1.8.5/intel padb/3.3
...
------------------------- /apps/modulefiles/libraries -------------------------
atlas/3.11.34(default)          netcdf/3.6.3/intel
...

----------------------- /apps/modulefiles/applications ------------------------
abinit/7.10.4(default)          namd/2.10/hybrid/normal
...
```

Εικόνα 2.3: Module avail command

Για να φορτώσετε ένα module, για παράδειγμα, τη GNU 5.1.0 compiler suite εκτελέστε: Module Load

```
module load gnu/5.1.0

gcc --version
gcc (GCC) 5.1.0
```

Εικόνα 2.4: Module load command I

Σημειώστε ότι κάθε module έχει ένα όνομα και την έκδοση. Η κλίση μόνο με το όνομα υποδηλώνει τη φόρτωση της προεπιλεγμένης έκδοσης module.

```
module load gnu

gcc --version
gcc (GCC) 4.9.2
```

Εικόνα 2.5: Module load command II

Για να ελέγξετε τα φορτωμένα modules εκτελέστε: module list

```
module list

Currently Loaded Modulefiles:
  1) gnu/4.9.2
```

Εικόνα 2.6: Module list command

Για να αφαιρέσετε το φορτωμένο module GNU εκτελέστε: module unload

```
module unload gnu/4.9.2
```

Εικόνα 2.7: Module unload command

Για να καθαρίσετε το περιβάλλον σας από όλα τα modules που έχουν φορτωθεί εκτελέστε: module purge

```
module purge
```

```
module list

No Modulefiles Currently Loaded.
```

Εικόνα 2.8: Module purge/list commands

Για εναλλαγή ανάμεσα σε module versions, για παράδειγμα, χρήση του GNU 5.1.0, αντί του φορτωμένου GNU 4.9.2 εκτελέστε: module switch

```
module switch gnu/5.1.0 gnu/4.9.2
```

Εικόνα 2.9: Module switch command

Για να ελέγξετε τις πληροφορίες σχετικά με τις αλλαγές του περιβάλλοντος που θα συμβούν αν φορτωθεί για παράδειγμα το module gnu/4.9.2 εκτελέστε: module show

```
module show gnu/4.9.2

-------------------------------------------------------------------
/apps/modulefiles/compilers/gnu/4.9.2:

module-whatis  Enable usage for the GNU Compiler Collection version 4.9.2
setenv     COMPILER_GNUROOT /apps/compilers/gnu/4.9.2
prepend-path   PATH /apps/compilers/gnu/4.9.2/bin
prepend-path   INCLUDE /apps/compilers/gnu/4.9.2/include
prepend-path   LD_LIBRARY_PATH /apps/compilers/gnu/4.9.2/lib
prepend-path   LD_LIBRARY_PATH /apps/compilers/gnu/4.9.2/lib64
prepend-path   MANPATH /apps/compilers/gnu/4.9.2/share/man
-------------------------------------------------------------------
```

Εικόνα 2.10: Module show command

Για να δείτε πληροφορίες για τα modules εκτελέστε: module whatis

```
module whatis gnu/4.9.2

gnu/4.9.2 : Enable usage for the GNU Compiler Collection version 4.9.2
```

Εικόνα 2.11: Module whatis command

Περισσότερη βοήθεια: module help

```
module help gnu/4.9.2
```

module sub-commands overview

Module Command	Description
avail	List all available modules.
load	Load module into the shell environment.
unload	Remove module from the shell environment.
list	List loaded modules.
purge	Unload all loaded modules.
switch module1 module2	Switch loaded module1 with module2.
show	List all of the environment changes the module will make if loaded
whatis	Display what is the module information
help	More specific help
keyword *string*	Seeks through the 'whatis' informations of all modules for the specified string.

Εικόνα 2.12: Module help command

Πλήρης κατάλογος των εντολών module στο man page.

```
man module
```

Εικόνα 2.13: Man module command

- **Επομένως εάν θέλετε να τρέξετε ένα από τα διαθέσιμα προγράμματα του ARIS (τα οποία υπάρχουν στο url:http://doc.aris.grnet.gr/software/) για παράδειγμα το ABYSS εκτελείτε στο περιβάλλον σας τη εντολή: module load abyss και στη συνέχεια τρέχετε το εκτελέσιμο με : ABYSS**

2.5. Εκτέλεση Προγραμμάτων στον ARIS

Όπως είπαμε ο ARIS χρησιμοποιεί ένα σύστημα batch για να κατανέμει δίκαια τους πόρους του στους χρήστες. Επομένως δεν μπορείτε να εκτελέσετε κάποιο πρόγραμμα (είτε δικό σας είτε του συστήματος) στους nodes του backend απλά τρέχοντας το εκτελέσιμο. Τα προγράμματα σας τα εκτελείτε δημιουργώντας jobs και καταθέτοντας τα στο σύστημα batch του ARIS το οποίο συγκεκριμένα είναι ο SLURM(Simple Linux Utility for Resource Management). Ο SLURM Workload Manager είναι ένα λογισμικό με πολλές λειτουργίες.Σε γενικές γραμμές:

- Δίνει αποκλειστική ή και μη-αποκλειστική πρόσβαση σε πόρους (compute/backend nodes) στους χρήστες για κάποιο χρονικό διάστημα ώστε να μπορούν να εκτελέσουν ένα job.

- Παρέχει ένα framework για την έναρξη, εκτέλεση και παρακολούθηση των εργασιών (συνήθως μια παράλληλη εργασία) σε ένα σύνολο από backend nodes.

- Κατανέμει τους πόρους και διαχειρίζεται την ουρά(queue) των pending jobs.

Ορολογία:

- nodes: οι nodes που χρησιμοποιεί το SLURM.Οι backend/compute nodes.
- partitions : οι nodes που χρησιμοποιεί το SLURM είναι χωρισμένοι σε ομάδες
- jobs: εργασίες που κατοχυρώνουν πόρους του συστήματος για την εκτέλεση των προγραμμάτων σας. Περιγράφονται με scripts.

Για να εκτελέσουμε ένα δικό μας πρόγραμμα λοιπόν ακολουθούμε τα παρακάτω βήματα:

1. Μετακινούμαστε στο προσωπικό μας φάκελο $HOME με cd $HOME όπου και έχουμε τον πηγαίο κώδικα.

2. Μεταγλωττίζουμε το κώδικα και μεταφέρουμε το εκτελέσιμο στο προσωπικό μας φάκελο του /work file system , $WORKDIR με mv [runnable.out] $WORKDIR .

3. Μετακινούμαστε στο φάκελο $WORKDIR με cd $WORKDIR.

4. Δημιουργούμε ένα script με nano [όνομα του script] .Το script αυτό θα περιγράφει το job που θα καταχωρίσουμε στη συνέχεια στο SLURM για να εκτελεστεί το πρόγραμμά μας στους backend nodes του ARIS.

Σύνταξη script

- **Script για σειριακό Πρόγραμμα** (**Προσοχή**! Διαβάστε τις επεξηγήσεις γιατί το script είναι γενικό και θα χρειαστεί να αλλάξετε τις παραμέτρους για να τρέξει το πρόγραμμα σας):

```
#!/bin/bash

#-----------------------------------------------------------------
# Serial job , requesting 1 core , 2800 MB of memory per job
#-----------------------------------------------------------------

#SBATCH --job-name=seraljob# Job name
#SBATCH --output=serialjob.%j.out # Stdout (%j expands to jobId)
#SBATCH --error=serialjob.%j.err # Stderr (%j expands to jobId)
#SBATCH --ntasks=1 # Total number of tasks
#SBATCH --nodes=1 # Total number of nodes requested
#SBATCH --ntasks-per-node=1 # Tasks per node
#SBATCH --cpus-per-task=1 # Threads per task
#SBATCH --mem=2800 # Memory per job in MB
#SBATCH -t 01:30:00 # Run time (hh:mm:ss) - (max 48h)
#SBATCH -A testproj # Accounting project

# Load any necessary modules
module load gnu
module load intel

# Launch the executable a.out
./runnable.out #ARGS
```

Επεξήγηση:

- Ότι ξεκινάει με #SBATCH είναι παράμετροι που χρησιμοποιεί το SLURM για τη διαχείριση των πόρων.
- Στη παράμετρο #SBATCH —job-name δίνουμε ένα όνομα στο job για να μπορούμε να ξεχωρίζουμε τα jobs σε άλλες λειτουργίες του SLARM που είναι για monitoring έλεγχο κτλ.
- Στη παράμετρο #SBATCH —output δίνουμε το όνομα του αρχείου που θα δημιουργηθεί αφού τερματίσει το πρόγραμμα μας και θα έχει τα αποτελέσματα.Το όνομα που του δίνει το συγκεκριμένο script περιλαμβάνει και το id του job.
- Στη παράμετρο #SBATCH —error δίνουμε το όνομα του αρχείου που θα δημιουργηθεί αφού τερματίσει το πρόγραμμα μας και θα έχει τυχόν errors.
- Στη παράμετρο #SBATCH —ntasks καθορίζεται πόσες διεργασίες θα τρέξουν το πρόγραμμά σας.Στη προκειμένη περίπτωση είναι μια η διεργασία γιατί το πρόγραμμα είναι σειριακό.
- Στη παράμετρο #SBATCH —nodes καθορίζεται ο αριθμός των backend nodes που θα τρέξουν το πρόγραμμα μας.Στη προκείμενη περίπτωση είναι ένας γιατί το πρόγραμμα μας θα το τρέξει μόνο μια διεργασία.
- Στη παράμετρο #SBATCH --ntasks-per-node καθορίζεται ο αριθμός των διεργασιών ανά node

- Στη παράμετρο #SBATCH --cpus-per-task καθορίζεται ο αριθμός των cpu ανά διεργασία σε περίπτωση που το πρόγραμμα είναι multithreaded.
- Στη παράμετρο #SBATCH —mem καθορίζεται η συνολική μνήμη που θα χρειαστεί το πρόγραμμά σας.
- Στη παράμετρο #SBATCH -t καθορίζεται ο συνολικός χρόνος που θα χρειαστεί το πρόγραμμά σας.
- Στη παράμετρο #SBATCH -A δίνουμε το όνομα του project για το accounting που μας έχουν δώσει οι διαχειριστές.Για να βρούμε το όνομα του project για το accounting εκτελούμε την εντολή ‘mybudget’ .

- **Script για MPI Πρόγραμμα:**

```
#!/bin/bash

#----------------------------------------------------------------
# Pure MPI job , using 80 procs on 4 nodes ,
# with 20 procs per node and 1 thread per MPI task

#----------------------------------------------------------------

#SBATCH --job-name=mpijob # Job name
#SBATCH --output=mpijob.%j.out # Stdout (%j expands to jobId)
#SBATCH --error=mpijob.%j.err # Stderr (%j expands to jobId)
#SBATCH --ntasks=80 # Total number of tasks
#SBATCH --nodes=4 # Total number of nodes requested
#SBATCH --ntasks-per-node=20 # Tasks per node
#SBATCH --cpus-per-task=1 # Threads per task(=1) for pure MPI
#SBATCH --mem=56000 # Memory per job in MB
#SBATCH -t 01:30:00 # Run time (hh:mm:ss) - (max 48h)
#SBATCH -A testproj # Accounting project

# Load any necessary modules

module load gnu
module load intel
module load intelmpi

export I_MPI_FABRICS=shm:dapl

# Launch the executable

srun EXE ARGS
```

- **Script για MPI/OPENMP Πρόγραμμα**:

```
#!/bin/bash

#----------------------------------------------------------------
# Hybrid MPI/OpenMP job , using 80 procs on 4 nodes ,
# with 2 procs per node and 10 threads per MPI task.
#----------------------------------------------------------------

#SBATCH --job-name=hybridjob # Job name
#SBATCH --output=hybridjob.%j.out # Stdout (%j expands to jobId)
#SBATCH --error=hybridjob.%j.err # Stderr (%j expands to jobId)
#SBATCH --ntasks=8 # Total number of tasks
#SBATCH --nodes=4 # Total number of nodes requested
#SBATCH --ntasks-per-node=2 # Tasks per node
#SBATCH --cpus-per-task=10 # Threads per task
#SBATCH --mem =56000 # Memory per job in MB
#SBATCH -t 01:30:00 # Run time (hh:mm:ss) - (max 48h)
#SBATCH -A testproj # Accounting project

# Load any necessary modules

module load gnu
module load intel
module load intelmpi

export I_MPI_FABRICS=shm:dapl

export OMP_NUM_THREADS=$SLURM_CPUS_PER_TASK

# Launch the executable
srun EXE ARGS
```

5. Αφού παραμετροποιήσουμε σωστά το script το αποθηκεύουμε και στη συνέχεια τρέχουμε την εντολή sbatch [όνομα του script] έτσι ώστε να καταχωρηθεί το job στο SLURM και να μπει σε ουρά προτεραιότητας.

6.Όταν εκτελεστεί τελικά το πρόγραμμά μας θα δημιουργηθούν τα δυο αρχεία .out και .err από τα οποία μπορούμε να δούμε τα αποτελέσματα.

*Ο ARIS δεν υποστηρίζει μέχρι στιγμής interactive jobs.Αυτό σημαίνει ότι τα προγράμματα σας δεν θα πρέπει να έχουν αλληλεπίδραση με το χρήστη κατά τη διάρκεια εκτέλεσης τους ζητώντας για παράδειγμα κάποιο input από το χρήστη.

*Τα jobs μπαίνουν σε ουρά αναμονής.Αυτό σημαίνει ότι το πρόγραμμα και κατ' επέκταση τα αποτελέσματα θα βγουν με κάποια καθυστέρηση.Κάποιοι από τους λόγους καθυστέρησης μπορεί να είναι οι παρακάτω:

Dependency	This job is waiting for a dependent job to complete.
NodeDown	A node required by the job is down.
PartitionDown	The partition (queue) required by this job is in a DOWN state and temporarily accepting no jobs, for instance because of maintenance. Note that this message may be displayed for a time even after the system is back up.
Priority	One or more higher priority jobs exist for this partition or advanced reservation. Other jobs in the queue have higher priority than yours.
ReqNodeNotAvail	No nodes can be found satisfying your limits, for instance because maintenance is scheduled and the job can not finish before it
Reservation	The job is waiting for its advanced reservation to become available.
Resources	The job is waiting for resources (nodes) to become available and will run when Slurm finds enough free nodes.
SystemFailure	Failure of the SLURM system, a file system, the network, etc

Εικόνα 2.14: Λόγοι καθυστέρησης των jobs στο slurm

SLURM commands

- sacct παρέχει χρήσιμες πληροφορίες για τα jobs σχετικά μετά στοιχειά τους (jobid,jobname κλπ) και την κατάσταση που βρίσκονται (running, canceled, complete κλπ).
- sbatch χρησιμοποιείται για να υποβάλει αίτημα στο SLURM για εκτέλεση ενός job.Αν το αίτημα εγκριθεί, το job μπαίνει στην ουρά.
- scancel χρησιμοποιείται για τη ματαίωση ενός job με state (pending, running). Μπορεί επίσης να χρησιμοποιηθεί για να στείλει σήμα σε όλες τις διεργασίες που συνδέονται με το job.
- scontrol είναι διαχειριστικό εργαλείο που χρησιμοποιείται για να δούμε ή να τροποποιήσουμε τη κατάσταση του SLURM. Κάποιες εντολές μπορούν να εκτελεστούν μόνο από τον root user(admin).
- sinfo αναφέρει την κατάσταση των partitions και nodes που διαχειρίζεται το SLURM. Έχει διάφορες εντολές φιλτραρίσματος, ταξινόμησης και μορφοποίησης.
- squeue αναφέρει την κατάσταση των jobs. Περιέχει πολλές εντολές φιλτραρίσματος, ταξινόμησης κλπ. Από προεπιλογή εμφανίζει τα running jobs και τα pending jobs με σειρά προτεραιότητας.

- srun χρησιμοποιείται για να υποβάλλεται μια εργασία για εκτέλεση. Η srun έχει μια μεγάλη ποικιλία από επιλογές για να καθορίσετε τις απαιτήσεις των πόρων, όπως: ελάχιστο και μέγιστο πλήθος nodes, αριθμό cpu's, συγκεκριμένων nodes για χρήση ή μη χρήση, και ειδικά χαρακτηριστικά του node (συγκεκριμένη μνήμη, χώρο στο δίσκο κ.λπ.). Ένα job μπορεί να περιέχει πολλαπλά στάδια (job steps) τα οποία εκτελούνται διαδοχικά ή παράλληλα.

Άλλες χρήσιμες Εντολές

mybudget: Check allocated core hours

```
$ mybudget
============================================================================
Core Hours Allocation Information for account : testproj
============================================================================
Allocated Core Hours   :       2400000.00
Consumed  Core Hours   :            15.00
Percentage of Consumed :             0.00
============================================================================
```

myreport: Check consumed core hours and energy.

```
$ myreport
-------------------------------------------------------------------------------
Cluster/Account/User Utilization 2015-04-07T00:00:00 - 2015-10-07T23:59:59 (15897600 secs)
Time reported in CPU Hours
-------------------------------------------------------------------------------
  Cluster         Account     Login     Proper Name       Used     Energy
--------- --------------- --------- --------------- ---------- ----------
     aris        testproj username       User Name         15        110
```

Το SLURM έχει πολλές ακόμα λειτουργίες που δεν αναφερθήκαμε αναλυτικά και είναι πολύ σημαντικές για τον έλεγχο την παρακολούθηση και τη διαχείριση των job σας.Μπορείτε να διαβάσετε γι αυτές αναλυτικότερα στους παρακάτω συνδέσμους:

http://doc.aris.grnet.gr/run/
https://computing.llnl.gov/linux/slurm/quickstart.html

2.6. Λογισμικό Απόδοσης Παράλληλων Προγραμμάτων

mpiP

- Το mpiP είναι πρόγραμμα μέτρησης απόδοσης για εφαρμογές MPI. Με τη χρήση του mpiP υπάρχει η δυνατότητα να βρεθούν το 90% των προβλημάτων ενός προγράμματος MPI καθώς παρέχονται πληροφορίες σχετικά με το πού παρατηρούνται καθυστερήσεις κατά την εκτέλεση του. Επειδή συλλέγει μόνο στατιστικές πληροφορίες για συναρτήσεις MPI, το mpiP επιβαρύνει σημαντικά λιγότερο σχετικά με tracing tools. Όλες οι πληροφορίες που συλλέγει το mpiP αφορούν τοπικές μετρήσεις κάθε διεργασίας. Το communication του MPI το χρησιμοποιεί μόνο για να συλλέξει τις πληροφορίες και να παράξει τα αποτελέσματα σε ένα ενιαίο αρχείο. Περισσότερες πληροφορίες για το mpiP στο παρακάτω σύνδεσμο: http://mpip.sourceforge.net

- Για να χρησιμοποιήσετε το mpiP στο πρόγραμμά σας στον ARIS μεταγλωττίστε το με τη παρακάτω εντολή:

```
mpicc -o  [executable]  [code.c] -L[mpiP_root}/lib] -lmpiP -lm -lbfd -liberty -lunwind
```

Όπου για να βρείτε το [mpiP_root}/lib] εκτελείτε την εντολή module show mpiP

- Αφού μεταγλωττίσετε το πρόγραμμα σας με το παραπάνω τρόπο το τρέχετε. Μετά την εκτέλεση τα αποτελέσματα των μετρήσεων θα εκτυπωθούν σε ένα αρχείο με όνομα παρόμοιο με αυτό: main.4.16473.1.mpiP

- Στο επόμενο κεφάλαιο θα αναλύσουμε και θα βγάλουμε συμπεράσματα από τα αποτελέσματα που μας δίνει η εκτέλεση του mpiP.

3. PROJECT ΣΤΟΝ ARIS

Στόχος του project που τρέξαμε στον ARIS ήταν να βελτιστοποιήσουμε, σε ένα δικό μας παράλληλο πρόγραμμα, τη κλιμάκωση του χρόνου εκτέλεσης σε συνάρτηση με το μέγεθος των δεδομένων. Παρακάτω παρουσιάζουμε το πρόγραμμά μας καθώς και το τρόπο με τον οποίο καταφέραμε να βελτιώσουμε το κώδικα και τέλος δίνουμε κάποιες γραφικές παραστάσεις με την απόδοση του αρχικού και του βελτιωμένου κώδικα.

3.1. Πρόγραμμα συνέλιξης εικόνας

Το πρόγραμμά μας υλοποιεί ένα φίλτρο συνέλιξης για αντιταύτιση και εξομάλυνση εικόνων σε RAW μορφή (εικόνες που δεν έχουν υποστεί συμπίεση). Δέχεται σαν είσοδο την εικόνα, τις διαστάσεις και το είδος της (έγχρωμη/ασπρόμαυρη) και την επεξεργάζεται σύμφωνα με ένα διδιάστατο διακριτό φίλτρο 3x3. Έτσι, διακριτή συνέλιξη μιας εικόνας εισόδου I_I με το φίλτρο *h'* για την παραγωγή μιας εικόνας εξόδου I_O ορίζεται ως εξής:

$$I_O(i,j) = \sum_{p=-s}^{s} \sum_{q=-s}^{s} I_I(i-p, j-q) * h(p,q)$$

όπου το διδιάστατο διακριτό φίλτρο *h* 'είναι κανονικοποιημένο και είναι της μορφής (για *s*=1):

$$h' = \begin{matrix} \frac{1}{16} & \frac{2}{16} & \frac{1}{16} \\ \frac{2}{16} & \frac{4}{16} & \frac{2}{16} \\ \frac{1}{16} & \frac{2}{16} & \frac{1}{16} \end{matrix}$$

Τα i και j καθορίζουν μοναδικά το pixel που δέχεται την επεξεργασία κάθε φορά και οι τιμές του διακριτού φίλτρου h' είναι συντελεστές του pixel αυτού και των γειτονικών του. Η συνέλιξη εφαρμόζεται πολλαπλές φορές πάνω στην εικόνα. Το πόσες φορές δίνεται σαν είσοδος.Επίσης ελέγχεται σύγκλιση (όταν δεν έχει μεταβληθεί η εικόνα μετά την εφαρμογή του φίλτρου) μετά από κάποιο αριθμό επαναλήψεων.

Η επεξεργασία των pixel γίνεται παράλληλα με mpi. Πιο συγκεκριμένα τα pixel της εικόνας χωρίζονται σε ομάδες οι οποίες επεξεργάζονται παράλληλα από διαφορετικές διεργασίες.Για την ταχύτερη επικοινωνία μεταξύ των διεργασιών χρησιμοποιείται persistent comunication και cartesian topology (mpi τεχνολογίες).

Παρακάτω παραθέτουμε των κώδικα. Για να το τρέξετε στον ARIS αφού το έχετε μεταγλωτίσει με “mpicc -o executeble main.c -lm” εκτελείτε :

srun ./executable -i image_path/image.raw -o new_image_path/out.raw -h (height) -w (width) -n repeats -c (3 for color, 1 for b/w)

```c
/*
 * TODO:
 * - remove verbose stuff
 */

/*
 * example usage:
 * mpicc mpi-parallel-io-convolution-multichannel-pers.c -std=c99 -lm -O2
 * mpiexec -f machines -np 36 ./a.out -o out.raw -i psproj/photos/landscape_w5040xh3840.raw -w 5040 -h 3840 -c 3 -n 20
 */

#include <sys/stat.h>
#include <stdarg.h>
#include <stdlib.h>
#include <string.h>
#include <limits.h>
#include <errno.h>
#define _GNU_SOURCE
#include <stdio.h>
#include <math.h>
#include "mpi.h"

#define MAX_PATH_LEN    (int) 256
#define master  0
#define true    1
#define false   0

/* Weighted 3x3 smoothing kernel with Gaussian blur */
const static int radius = 3;
const static int kernel[3][3] = {
    { 0, -1, 0 },
    { -1, 5, -1},
    { 0, -1, 0 }
};
static int sum = 0;

struct args;
static void get_args(int argc, char **argv, struct args *args, int size);
static void broadcast_args(struct args *args);

struct args {
    int height, width, iterations, channels;
    char input[MAX_PATH_LEN], output[MAX_PATH_LEN];
};


static void get_args(int argc, char **argv, struct args *args, int size){
    if(argc != 13){
    abort_and_show_usage:
        fprintf(stderr, "usage:  mpiexec [-f machines] -np processes " \
                "%s -i input -o output -h height -w width " \
                "-n iterations -c channels\n", *argv);
        MPI_Abort(MPI_COMM_WORLD, 0);
    }
    int i;
    for(i=1; i<argc; i+=2)
        if(argv[i][0] != '-')
            goto abort_and_show_usage;
        else switch(argv[i][1]){
            case 'i': strncpy(args->input, argv[i+1], sizeof(args->input));
                break;
            case 'o': strncpy(args->output, argv[i+1], sizeof(args->output));
                break;
            case 'h': args->height = atoi(argv[i+1]); break;
            case 'w': args->width = atoi(argv[i+1]); break;
            case 'n': args->iterations = atoi(argv[i+1]); break;
            case 'c': args->channels = atoi(argv[i+1]); break;
            default: goto abort_and_show_usage;
        }
    char *p;
    if((p = strrchr(args->input,'.')) == NULL || strcmp(p+1,"raw") ||
       (p = strrchr(args->output,'.')) == NULL || strcmp(p+1,"raw")){
        fprintf(stderr,"only raw format is supported.\n");
        MPI_Abort(MPI_COMM_WORLD, 0);
    }
    struct stat st;
    if(stat(args->input, &st) < 0 || args->height < 0 || args->width < 0 ||
       args->iterations < 0 || (args->channels != 1 && args->channels != 3))
        goto abort_and_show_usage;
    if(args->height * args->width % size || sqrt(size) * sqrt(size) != size){
        fprintf(stderr,"image dimensions not divisible by process topology.\n");
        MPI_Abort(MPI_COMM_WORLD, 0);
    }
```

```
}

static void inline convolve(unsigned char *data,unsigned char *buffer,int index,int bheight,int bwidth, int channel, int channels){
    int value = data[(index-bwidth-3) * channels+ channel] * kernel[0][0] +    /* up left */
    data[(index-bwidth-2) * channels+ channel] * kernel[0][1] + /* up */
    data[(index-bwidth-1) * channels+ channel] * kernel[0][2] + /* up right */
    data[(index-1) * channels+ channel] * kernel[1][0] +       /* left */
    data[(index) * channels+ channel] * kernel[1][1] +         /* middle */
    data[(index+1) * channels+ channel] * kernel[1][2] +       /* right */
    data[(index+bwidth+1) * channels+ channel] * kernel[2][0] + /* down left */
    data[(index+bwidth+2) * channels+ channel] * kernel[2][1] + /* down */
    data[(index+bwidth+3) * channels+ channel] * kernel[2][2];  /* down right */
    value /= sum;
    buffer[index* channels+ channel] = value > UCHAR_MAX ? UCHAR_MAX : value < 0 ? 0 : value;
}

static void broadcast_args(struct args *args){
    MPI_Datatype struct_datatype;
    int blocklengths[6] = { 1, 1, 1, 1, MAX_PATH_LEN, MAX_PATH_LEN };
    MPI_Aint displacements[6];
    /* compute displacements of structure components */
    MPI_Address(&args->height, &displacements[0]);
    MPI_Address(&args->width, &displacements[1]);
    MPI_Address(&args->iterations, &displacements[2]);
    MPI_Address(&args->channels, &displacements[3]);
    MPI_Address(&args->input, &displacements[4]);
    MPI_Address(&args->output, &displacements[5]);
    MPI_Aint base = displacements[0];
    int i;
    for(i=0; i<6; ++i)
        displacements[i] -= base;
    MPI_Datatype types[6] = { MPI_INT, MPI_INT, MPI_INT, MPI_INT, MPI_CHAR,
        MPI_CHAR };
    MPI_Type_struct(6, blocklengths, displacements, types, &struct_datatype);
    MPI_Type_commit(&struct_datatype);
    MPI_Bcast(args, 1, struct_datatype, master, MPI_COMM_WORLD);
    MPI_Type_free(&struct_datatype);
}


int main(int argc,char **argv){
    MPI_Init(&argc, &argv);
    int rank, size;
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
    MPI_Comm_size(MPI_COMM_WORLD, &size);

    struct args args;
    if(rank == master)
        get_args(argc, argv, &args, size);
    /* broadcast arguments */
    broadcast_args(&args);

    int height = args.height,
    width = args.width,
    iterations = args.iterations,
    channels = args.channels;
    if(rank == master)
        printf("info: input = %s output = %s height = %4d width = %4d " \
               "iterations = %4d channels = %s\n", args.input, args.output,
               height, width, iterations, channels != 1 ? "RGB" : "Grayscale");

    /* Calculate the kernel sum. If the sum of coefficients is 0 set it to 1 */
    int j;
    int i;
    for(i=0; i<radius; ++i)
        for(j=0; j<radius; ++j)
            sum += kernel[i][j];
    sum = !sum ? 1:sum;

    /*** cartesian grid topology ***/
    int ndims = 2, *dims;
    if((dims = calloc(ndims, sizeof(int))) == NULL)
        MPI_Abort(MPI_COMM_WORLD, 0);
    MPI_Dims_create(size, ndims, dims);  /* get cartesian grid dimensions */
    /* create the cartesian communicator */
    MPI_Comm comm;
    int periods[2] = {0, 0};    /* periodicity set to false on x,y dimensions */
    int reorder = 1;            /* ranking may be reordered by the topology */
    MPI_Cart_create(MPI_COMM_WORLD, ndims, dims, periods, reorder, &comm);
    /* get process rank on the cartesian grid */
    MPI_Comm_rank(comm, &rank);
    /* get process coordinates in the cartesian grid */
    int coords[2], ncoords[2];
    MPI_Cart_coords(comm, rank, ndims, coords);
    int neighbor[8];    /* array of neighbor ranks */
    for(i=0; i<8; ++i)
        neighbor[i] = MPI_PROC_NULL;    /* initialize to MPI_PROC_NULL */
    /* up-left neighbor [0] */
    if((ncoords[0]=coords[0]-1) >= 0 && (ncoords[1]=coords[1]-1) >= 0)
        MPI_Cart_rank(comm, ncoords, &neighbor[0]);
    /* up-right neighbor [2] */
    if((ncoords[0]=coords[0]-1) >= 0 && (ncoords[1]=coords[1]+1) < dims[1])
```

```
    /* down-left neighbor [5] */
    if((ncoords[0]=coords[0]+1) < dims[0] && (ncoords[1]=coords[1]-1) >= 0)
        MPI_Cart_rank(comm, ncoords, &neighbor[5]);
    /* down-right neighbor [7] */
    if((ncoords[0]=coords[0]+1) < dims[0] && (ncoords[1]=coords[1]+1) < dims[1])
        MPI_Cart_rank(comm, ncoords, &neighbor[7]);
    /* upper [1] and lower [6] neighbor */
    MPI_Cart_shift(comm, 0, 1, &neighbor[1], &neighbor[6]);
    /* left [3] and right [4] neighbor */
    MPI_Cart_shift(comm, 1, 1, &neighbor[3], &neighbor[4]);

    /*** cartesian topology info ***/
    if(rank == master)
        printf("info: cartesian grid dimensions = %dx%d\n", dims[0], dims[1]);

    /*** block-size assigned to each process ***/
    int bheight = height/dims[0]; /* block height */
    int bwidth = width/dims[1];   /* block width */

    /*** custom datatypes ***/
    /** subarray datatype (inner frame) **/
    MPI_Datatype subarray_datatype;
    int subarray_size[2];
    subarray_size[0] = height;  /* image height in MPI_UNSIGNED_CHAR */
    subarray_size[1] = width * channels;   /* image width in MPI_UNSIGNED_CHAR */
    int subarray_subsize[2];
    subarray_subsize[0] = bheight;  /* subarray height in MPI_UNSIGNED_CHAR */
    subarray_subsize[1] = bwidth * channels;   /* subarray width in MPI_UNSIGNED_CHAR */
    int subarray_displacement[2];
    /* height in which subarray starts in MPI_UNSIGNED_CHAR */
    subarray_displacement[0] = coords[0] * subarray_subsize[0];
    /* width in which subarray starts in MPI_UNSIGNED_CHAR */
    subarray_displacement[1] = coords[1] * subarray_subsize[1];
    MPI_Type_create_subarray(2, subarray_size, subarray_subsize,
                             subarray_displacement, MPI_ORDER_C, MPI_UNSIGNED_CHAR,
                             &subarray_datatype);
    MPI_Type_commit(&subarray_datatype);

    /** row datatype (upper and lower margin) **/
    /* count = bwidth, blocklength = 1, stride = 1 */
    MPI_Datatype row_datatype;
    MPI_Type_vector(bwidth, channels, channels, MPI_UNSIGNED_CHAR, &row_datatype);
    MPI_Type_commit(&row_datatype);

    /** column datatype (right and left margin) **/
    /* count = bheight, blocklength = 1, stride = bwidth+2 */
    MPI_Datatype col_datatype;
    MPI_Type_vector(bheight, channels, (bwidth+2)*channels, MPI_UNSIGNED_CHAR, &col_datatype);
    MPI_Type_commit(&col_datatype);

    /*** t, t+1 generation buffers ***/
    unsigned char *data = calloc((bheight+2) * (bwidth+2) * channels,
                                 sizeof(unsigned char));
    unsigned char *buffer = calloc((bheight+2) * (bwidth+2) * channels,
                                   sizeof(unsigned char));

    /*** MPI-IO parallel read ***/
    MPI_File fp;
    MPI_File_open(comm, args.input, MPI_MODE_RDONLY, MPI_INFO_NULL, &fp);
    MPI_File_set_view(fp, 0, MPI_UNSIGNED_CHAR, subarray_datatype, "native",
                      MPI_INFO_NULL);
    MPI_File_read_all(fp, data, subarray_subsize[0] * subarray_subsize[1], MPI_UNSIGNED_CHAR,
                      MPI_STATUS_IGNORE);
    MPI_File_close(&fp);

    // ------------------------------------------------------------------------

    /* rearrange buffer (add halo cells) */
    for(i=bheight-1; i>=0; --i)
        memmove(&data[((i+1)*(bwidth+2)+1)*channels], &data[i*bwidth*channels],
                bwidth*channels*sizeof(unsigned char));

    /* for process that have no neighbor, replicate their margins */
    /* replicate upper margin */
    if(neighbor[1] != MPI_PROC_NULL)
        memmove(&data[channels], &data[(bwidth+3)*channels],
                bwidth*channels*sizeof(unsigned char));
    /* replicate lower margin */
    if(neighbor[6] != MPI_PROC_NULL)
        memmove(&data[((bheight+1)*(bwidth+2)+1)*channels],
                &data[(bheight*(bwidth+2)+1)*channels],
                bwidth*channels*sizeof(unsigned char));
    /* replicate upper left corner */
    if(neighbor[0] != MPI_PROC_NULL)
        memmove(&data[channels], &data[((bwidth+2)+1)*channels],
                channels*sizeof(unsigned char));
    /* replicate upper right corner */
    if(neighbor[2] != MPI_PROC_NULL)
        memmove(&data[(bwidth+1)*channels], &data[(2*(bwidth+2)-2)*channels],
                channels*sizeof(unsigned char));
    /* replicate lower left corner */
    if(neighbor[5] != MPI_PROC_NULL)
```

```
        memmove(&data[(bheight+1)*(bwidth+2)*channels],
                &data[(bheight*(bwidth+2)+1)*channels], channels*sizeof(unsigned char));
    /* replicate lower right corner */
    if(neighbor[7] != MPI_PROC_NULL)
        memmove(&data[((bheight+2)*(bwidth+2)-1)*channels],
                &data[((bheight+1)*(bwidth+2)-2)*channels],
                channels*sizeof(unsigned char));
    /* replicate left margin */
    if(neighbor[3] != MPI_PROC_NULL)
        for(i=0; i<bheight; ++i)
            memmove(&data[(i+1)*(bwidth+2)*channels],
                    &data[((i+1)*(bwidth+2)+1)*channels], channels*sizeof(unsigned char));
    /* replicate right margin */
    if(neighbor[4] != MPI_PROC_NULL)
        for(i=0; i<bheight; ++i)
            memmove(&data[((i+1)*(bwidth+2)+bwidth+1)*channels],
                    &data[((i+1)*(bwidth+2)+bwidth)*channels],
                    channels*sizeof(unsigned char));

    MPI_Request request[32];
    /** persistent communication **/
    if(neighbor[0] != MPI_PROC_NULL){
        MPI_Send_init(&data[(bwidth+3)*channels], channels, MPI_UNSIGNED_CHAR,
                      neighbor[0], 0, comm, &request[0]);
        MPI_Recv_init(&data[0], channels, MPI_UNSIGNED_CHAR, neighbor[0], 7,
                      comm, &request[1]);
        MPI_Send_init(&buffer[(bwidth+3)*channels], channels, MPI_UNSIGNED_CHAR,
                      neighbor[0], 0, comm, &request[16]);
        MPI_Recv_init(&buffer[0], channels, MPI_UNSIGNED_CHAR, neighbor[0], 7,
                      comm, &request[17]);
    }
    if(neighbor[1] != MPI_PROC_NULL){
        MPI_Send_init(&data[(bwidth+3)*channels], 1, row_datatype, neighbor[1],
                      1, comm, &request[2]);
        MPI_Recv_init(&data[1 * channels], 1, row_datatype, neighbor[1], 6,
                      comm, &request[3]);
        MPI_Send_init(&buffer[(bwidth+3)*channels], 1, row_datatype, neighbor[1],
                      1, comm, &request[18]);
        MPI_Recv_init(&buffer[1 * channels], 1, row_datatype, neighbor[1], 6,
                      comm, &request[19]);
    }
    if(neighbor[2] != MPI_PROC_NULL){
        MPI_Send_init(&data[(2*(bwidth+2)-2)*channels], channels,
                      MPI_UNSIGNED_CHAR, neighbor[2], 2, comm, &request[4]);
        MPI_Recv_init(&data[(bwidth+1)*channels], channels, MPI_UNSIGNED_CHAR,
                      neighbor[2], 5, comm, &request[5]);
        MPI_Send_init(&buffer[(2*(bwidth+2)-2)*channels], channels,
                      MPI_UNSIGNED_CHAR, neighbor[2], 2, comm, &request[20]);
        MPI_Recv_init(&buffer[(bwidth+1)*channels], channels, MPI_UNSIGNED_CHAR,
                      neighbor[2], 5, comm, &request[21]);
    }
    if(neighbor[3] != MPI_PROC_NULL){
        MPI_Send_init(&data[(bwidth+3)*channels], 1, col_datatype, neighbor[3],
                      3, comm, &request[6]);
        MPI_Recv_init(&data[(bwidth+2)*channels], 1, col_datatype, neighbor[3],
                      4, comm, &request[7]);
        MPI_Send_init(&buffer[(bwidth+3)*channels], 1, col_datatype, neighbor[3],
                      3, comm, &request[22]);
        MPI_Recv_init(&buffer[(bwidth+2)*channels], 1, col_datatype, neighbor[3],
                      4, comm, &request[23]);
    }
    if(neighbor[4] != MPI_PROC_NULL){
        MPI_Send_init(&data[(2*(bwidth+2)-2)*channels], 1, col_datatype,
                      neighbor[4], 4, comm, &request[8]);
        MPI_Recv_init(&data[(2*(bwidth+2)-1)*channels], 1, col_datatype,
                      neighbor[4], 3, comm, &request[9]);
        MPI_Send_init(&buffer[(2*(bwidth+2)-2)*channels], 1, col_datatype,
                      neighbor[4], 4, comm, &request[24]);
        MPI_Recv_init(&buffer[(2*(bwidth+2)-1)*channels], 1, col_datatype,
                      neighbor[4], 3, comm, &request[25]);
    }
    if(neighbor[5] != MPI_PROC_NULL){
        MPI_Send_init(&data[(bheight*(bwidth+2)+1)*channels], channels,
                      MPI_UNSIGNED_CHAR, neighbor[5], 5, comm, &request[10]);
        MPI_Recv_init(&data[(bheight+1)*(bwidth+2)*channels], channels,
                      MPI_UNSIGNED_CHAR, neighbor[5], 2, comm, &request[11]);
        MPI_Send_init(&buffer[(bheight*(bwidth+2)+1)*channels], channels,
                      MPI_UNSIGNED_CHAR, neighbor[5], 5, comm, &request[26]);
        MPI_Recv_init(&buffer[(bheight+1)*(bwidth+2)*channels], channels,
                      MPI_UNSIGNED_CHAR, neighbor[5], 2, comm, &request[27]);
    }
    if(neighbor[6] != MPI_PROC_NULL){
        MPI_Send_init(&data[(bheight*(bwidth+2)+1)*channels], 1, row_datatype,
                      neighbor[6], 6, comm, &request[12]);
        MPI_Recv_init(&data[((bheight+1)*(bwidth+2)+1) * channels], 1,
                      row_datatype, neighbor[6], 1, comm, &request[13]);
        MPI_Send_init(&buffer[(bheight*(bwidth+2)+1)*channels], 1, row_datatype,
                      neighbor[6], 6, comm, &request[28]);
        MPI_Recv_init(&buffer[((bheight+1)*(bwidth+2)+1) * channels], 1,
                      row_datatype, neighbor[6], 1, comm, &request[29]);
    }
    if(neighbor[7] != MPI_PROC_NULL){
```

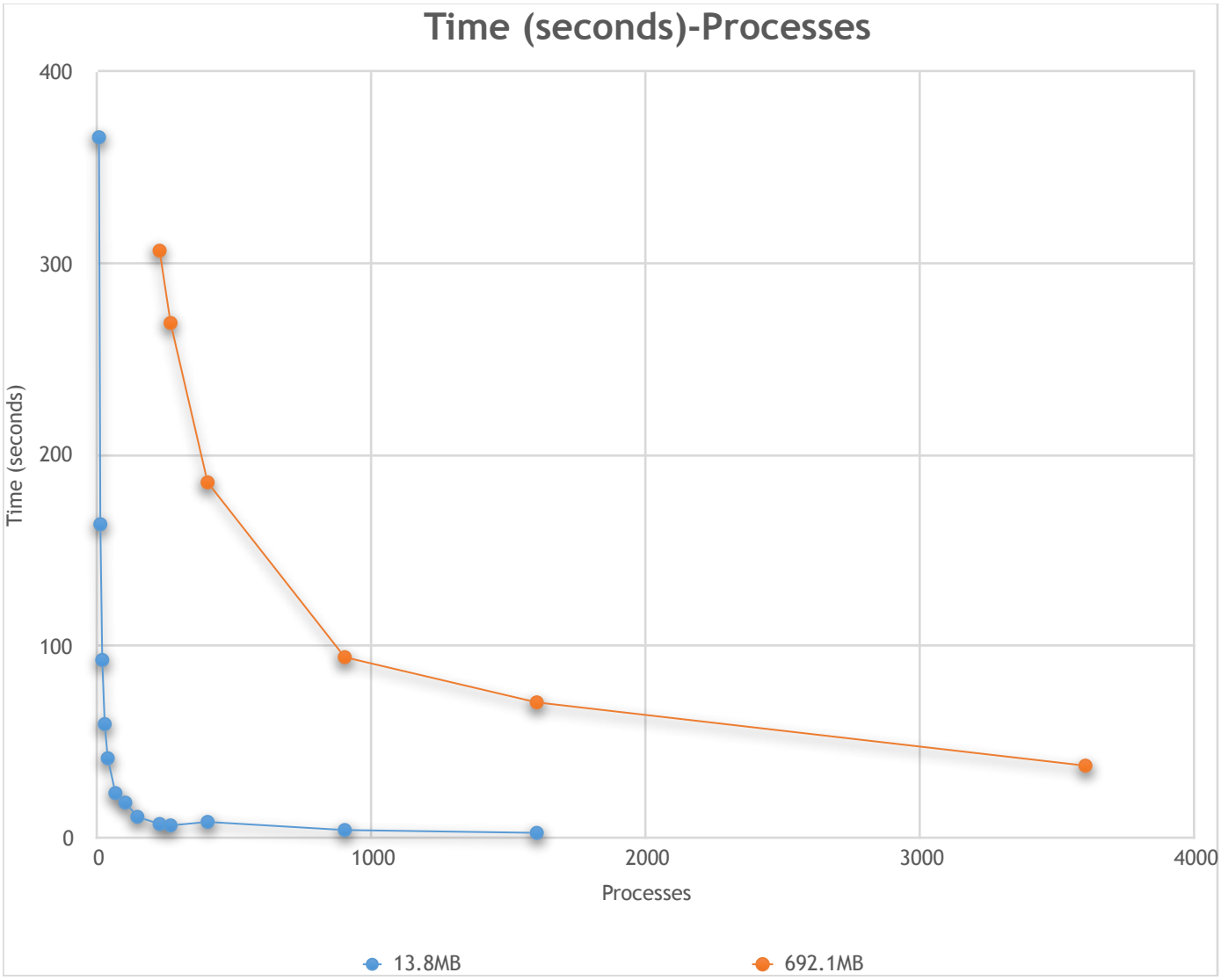
```
        MPI_Send_init(&data[((bheight+1)*(bwidth+2)-2)*channels], channels,
                      MPI_UNSIGNED_CHAR, neighbor[7], 7, comm, &request[14]);
        MPI_Recv_init(&data[((bheight+2)*(bwidth+2)-1)*channels], channels,
                      MPI_UNSIGNED_CHAR, neighbor[7], 0, comm, &request[15]);
        MPI_Send_init(&buffer[((bheight+1)*(bwidth+2)-2)*channels], channels,
                      MPI_UNSIGNED_CHAR, neighbor[7], 7, comm, &request[30]);
        MPI_Recv_init(&buffer[((bheight+2)*(bwidth+2)-1)*channels], channels,
                      MPI_UNSIGNED_CHAR, neighbor[7], 0, comm, &request[31]);
    }

    double ts,te,tes,tee;
    double local_computation_time = 0.0, local_communication_time = 0.0;
    int iter,offset,global_convergence = false,channel;
    unsigned char *temp;

    int check_freq = (int) sqrt(iterations);
    if(rank == master)
        printf("info: checking for convergence every %d iterations\n", check_freq);

    MPI_Barrier(comm);
    tes = MPI_Wtime();
    int local_iterations=0;
    /* #pragma omp parallel */
    for(iter=0; iter<iterations && !global_convergence; ++iter){

        offset = iter % 2 ? 16 : 0; /* iter %2 = 0:buffer, 1:data */
        ts = MPI_Wtime();
        if(neighbor[0] != MPI_PROC_NULL){
            MPI_Start(&request[0 + offset]);
            MPI_Start(&request[1 + offset]);
        }
        if(neighbor[1] != MPI_PROC_NULL){
            MPI_Start(&request[2 + offset]);
            MPI_Start(&request[3 + offset]);
        }
        if(neighbor[2] != MPI_PROC_NULL){
            MPI_Start(&request[4 + offset]);
            MPI_Start(&request[5 + offset]);
        }
        if(neighbor[3] != MPI_PROC_NULL){
            MPI_Start(&request[6 + offset]);
            MPI_Start(&request[7 + offset]);
        }
        if(neighbor[4] != MPI_PROC_NULL){
            MPI_Start(&request[8 + offset]);
            MPI_Start(&request[9 + offset]);
        }
        if(neighbor[5] != MPI_PROC_NULL){
            MPI_Start(&request[10 + offset]);
            MPI_Start(&request[11 + offset]);
        }
        if(neighbor[6] != MPI_PROC_NULL){
            MPI_Start(&request[12 + offset]);
            MPI_Start(&request[13 + offset]);
        }
        if(neighbor[7] != MPI_PROC_NULL){
            MPI_Start(&request[14 + offset]);
            MPI_Start(&request[15 + offset]);
        }
        te = MPI_Wtime();
        local_communication_time += te-ts;

        ts = te;
        for(channel=0; channel<channels; ++channel)
        /* #pragma omp for collapse(2) */
            for(i=0; i<bheight-2; ++i)
                for(j=0; j<bwidth-2; ++j)
                    convolve(data, buffer, (i+2)*(bwidth+2)+j+2, bheight,
                             bwidth, channel, channels);

        /* http://stackoverflow.com/questions/10540760/openmp-nested-parallel-for-loops-vs-inner-parallel-for */
        // #pragma omp parallel for default(none) shared(data, buffer, bheight, bwidth) schedule(static)
        // for(int k=0; k<(bheight-2)*(bwidth-2); ++k){
        //     int i=k/(bwidth-2);
        //     int j=k%(bwidth-2);
        //     convolve(data,buffer,(i+2)*(bwidth+2)+j+2,bheight,bwidth);
        // }

        te = MPI_Wtime();
        local_computation_time += te-ts;
        ts = te;
        for(i=0; i<8; ++i)
            if(neighbor[i] != MPI_PROC_NULL){
                MPI_Wait(&request[2*i + offset],MPI_STATUS_IGNORE);
                MPI_Wait(&request[2*i+1 + offset],MPI_STATUS_IGNORE);
            }
        te = MPI_Wtime();
        local_communication_time += te-ts;
        ts = te;
        /** convolve outer matrix **/
        /* convolve upper and lower margin elements */
```

```
452         for(channel=0; channel<channels; ++channel)
453             for(i=0; i<bwidth; ++i){
454                 convolve(data, buffer, bwidth+i+3, bheight, bwidth, channel, channels);    /* upper margin */
455                 convolve(data, buffer, bheight*(bwidth+2)+i+1, bheight, bwidth, channel, channels);    /* lower margin */
456             }
457         /* convolve left and right margin elements */
458         for(channel=0; channel<channels; ++channel)
459             for(i=0; i<bheight-2; ++i){
460                 convolve(data,buffer,(i+2)*(bwidth+2)+1,bheight,bwidth, channel, channels);    /* left margin */
461                 convolve(data,buffer,(i+3)*(bwidth+2)-2,bheight,bwidth, channel, channels);    /* right margin */
462             }
463         if(iter % check_freq == 0){
464             /** convergence check **/
465             int local_convergence = false;
466             for(i=0; i<bheight+2 && local_convergence; ++i)
467                 for(j=0; j<(bwidth+2) && local_convergence; ++j)
468                     for(channel=0; channel<channels; ++channel)
469                         if(data[(i*(bwidth+2)+j) * channels + channel] != buffer[(i*(bwidth+2)+j) * channels + channel])
470                             local_convergence = true;
471             MPI_Allreduce(&local_convergence, &global_convergence, 1, MPI_INT, MPI_MIN, comm);
472         }
473 
474         /* swap buffers */
475         temp = data;
476         data = buffer;
477         buffer = temp;
478 
479         local_computation_time += te-ts;
480         local_iterations++;
481     }
482     tee = MPI_Wtime();
483     double local_execution_time = tee-tes;
484 
485     /* reduce */
486     int total_iterations;
487     MPI_Reduce(&local_iterations, &total_iterations, 1, MPI_INT, MPI_SUM, master, comm);
488     double global_execution_time;
489     MPI_Reduce(&local_execution_time, &global_execution_time, 1, MPI_DOUBLE, MPI_MAX, master, comm);
490     double global_communication_time;
491     MPI_Reduce(&local_communication_time, &global_communication_time, 1, MPI_DOUBLE, MPI_MAX, master, comm);
492     double global_computation_time;
493     MPI_Reduce(&local_computation_time, &global_computation_time, 1, MPI_DOUBLE, MPI_MAX, master, comm);
494 
495     /* rearrange buffer (remove halo cells) */
496     for(i=0; i<bheight; ++i)
497         memmove(&data[i*bwidth*channels], &data[((i+1)*(bwidth+2)+1)*channels],
498                 bwidth*channels*sizeof(unsigned char));
499 
500     /*** MPI-IO parallel write ***/
501     MPI_File_open(comm, args.output, MPI_MODE_WRONLY | MPI_MODE_CREATE,
502                   MPI_INFO_NULL, &fp);
503     MPI_File_set_view(fp, 0, MPI_UNSIGNED_CHAR, subarray_datatype, "native",
504                       MPI_INFO_NULL);
505     MPI_File_write_all(fp, data, subarray_subsize[0]*subarray_subsize[1],
506                        MPI_UNSIGNED_CHAR, MPI_STATUS_IGNORE);
507     MPI_File_close(&fp);
508 
509     if(rank == master){
510         printf("\e[32m***\e[0m %-20s %d/%d \n", "iterations:", iter, iterations);
511         printf("\e[34m***\e[0m %-20s %lf sec\n", "execution time:", global_execution_time);
512         printf("\e[31m***\e[0m %-20s %lf sec\n", "communication time:", global_communication_time);
513         printf("\e[33m***\e[0m %-20s %lf sec\n", "computation time:", global_computation_time);
514         printf("\e[33m***\e[0m %-20s %d \n", "Total iterations of all processes:", total_iterations);
515     }
516 
517     /* deallocate the resources */
518     for(i=0; i<8; ++i)
519         if(neighbor[i] != MPI_PROC_NULL){
520             MPI_Request_free(&request[2*i]);
521             MPI_Request_free(&request[2*i+1]);
522             MPI_Request_free(&request[2*i+16]);
523             MPI_Request_free(&request[2*i+17]);
524         }
525     MPI_Type_free(&subarray_datatype);
526     MPI_Type_free(&row_datatype);
527     MPI_Type_free(&col_datatype);
528     MPI_Comm_free(&comm);
529     free(buffer);
530     free(data);
531     free(dims);
532     MPI_Finalize();
533 
534     exit(EXIT_SUCCESS);
535 }
```

Εικόνα 3.1: Το πρόγραμμά μας

Εκτελέσεις στον ARIS

- Εκτελέσαμε το πρόγραμμα για δυο rgb εικόνες διαφορετικού μεγέθους με διαφορετικό αριθμό διεργασιών κάθε φορά για να δούμε τη κλιμάκωση του. Η μία ήταν μεγέθους 14MB και width X height : 1920 X 2520. Η άλλη ήταν μεγέθους 700MB και width X height: 96000 X 2520.Η πράξη της συνέλιξης γινόταν 3000 φορές σε κάθε εκτέλεση και στις δύο περιπτώσεις.Τα αποτελέσματα φαίνονται στα διαγράμματα παρακάτω.



Σχήμα 1: Time-Processes

Αναλυτικότερα οι μετρήσεις είναι :

Πίνακας 1: Μετρήσεις χρόνων εκτέλεσης του προγράμματός μας στον A.R.I.S.

Processes	Time for 13.8MB image (seconds)	Time for 692.1MB image (seconds)
1	**1453,66138**	68018,477685
4	366.016821	
9	163.911496	
16	92.994998	
25	59.507241	
36	41.669567	
64	23.467865	
100	18.499604	
144	11.06766	
225	7.330908	306.777315
265	6.588478	269.024051
400	8.409939	185.758867
900	4.174377	94.449947
1600	2.697908	70.85403
3600		37.800881

- Από τη γραφική παράσταση παρατηρούμε ότι ο τρόπος κλιμάκωσης του χρόνου αναλόγως με τον αριθμό των διεργασιών είναι παρόμοιος και για τα δύο μεγέθη των εικόνων αν και για τη μεγαλύτερη εικόνα η κλιμάκωση δεν είναι το ίδιο καλή.Αυτό μπορούμε να το συμπεράνουμε παρατηρώντας ότι ο χρόνος για τη μικρότερη εικόνα πέφτει κατακόρυφα όσο αυξάνονται οι διεργασίες κάτι που δεν είναι τόσο αισθητό στη μεγαλύτερη εικόνα.Επομένως αφού το πρόγραμμά δεν κλιμακώνει κατά τον ίδιο τρόπο καθώς αυξάνουμε τα δεδομένα πιθανότατα να χρήζει βελτιστοποίησης του παράλληλου αλγορίθμου του κάτι το οποίο επιχειρούμε στο επόμενο κεφάλαιο.

3.2. Βελτίωση απόδοσης με τη βοήθεια του mpiP

Στην ενότητα 2.6 παρουσιάσαμε το mpiP το οποίο είναι ένα λογισμικό απόδοσης παράλληλων προγραμμάτων και δώσαμε οδηγίες για την εφαρμογή του.Αφού λοιπόν τρέξαμε το mpiP για το πρόγραμμά μας, αναλύσαμε τα αποτελέσματα που μας έδωσε στο αρχείο main.4.16473.1.mpiP. Το mpiP ανιχνεύει όλες τις mpi εντολές/συναρτήσεις που χρησιμοποιεί ένα παράλληλο πρόγραμμα και καταγράφει το μερίδιο του χρόνου που αντιστοιχεί στη κάθε μια σε σχέση με τον ολικό χρόνο εκτέλεσης του προγράμματος.Με αυτόν το τρόπο μπορέσαμε να εντοπίσουμε σε ποια σημεία καταναλώνει περισσότερο χρόνο το πρόγραμμά μας και να εξετάσουμε αν σ' αυτά τα σημεία μπορούν να γίνουν κάποιες αλλαγές.

Εάν κοιτάξουμε λοιπόν στο αρχείο που παραθέτουμε παρακάτω στη παράγραφο Aggregate Time (top twenty, descending, milliseconds) παρατηρούμε ότι η συνάρτηση AllReduce είναι αυτή που καταναλώνει το 55% του συνολικού mpi χρόνου. Αυτό σημαίνει ότι αν μπορούσαμε να την καταργήσουμε ή έστω να μειώσουμε τον αριθμό εμφάνισης της σίγουρα θα βελτίωνε την απόδοση του προγράμματος. Στη δικιά μας περίπτωση η ALLReduce (γραμμή 463-472 στο κώδικα) εμφανιζόταν πολλαπλές φορές στο κώδικα και καταφέραμε να τις ελαχιστοποιήσουμε χωρίς να επηρεαστεί το αποτέλεσμα.

Επίσης καταφέραμε να κάνουμε κάποιες ακόμα βελτιώσεις.Η δραστικότερη ήταν ο ορισμός δικών μας συναρτήσεων όπου καλούνται στο πρόγραμμα πολλαπλές φορές, ως inline (γραμμή 86 στο κώδικα). Άλλες βελτιώσεις όχι τόσο σημαντικές ήταν η αφαίρεση περιττών δηλώσεων και αρχικοποιήσεων που γίνονταν πολλαπλές φορές κατά τη διάρκεια εκτέλεσης.

main.4.16473.1.mpiP:

```
@ mpiP
@ Command : /work/pa001/parsysdi/convolution/./main -i ./image.raw -o ./out.raw -h
2520 -w 1920 -n 300 -c 3
@ Version                  : 3.4.1
@ MPIP Build date          : Sep  7 2015, 16:33:51
@ Start time               : 2015 12 16 22:21:05
@ Stop time                : 2015 12 16 22:21:41
@ Timer Used               : PMPI_Wtime
@ MPIP env var             : [null]
@ Collector Rank           : 0
@ Collector PID            : 19903
@ Final Output Dir         : .
@ Report generation        : Collective
@ MPI Task Assignment      : 0 node073
@ MPI Task Assignment      : 1 node073
@ MPI Task Assignment      : 2 node073
@ MPI Task Assignment      : 3 node073
---------------------------------------------------------------------------
@--- MPI Time (seconds) ---------------------------------------------------
---------------------------------------------------------------------------
Task    AppTime    MPITime     MPI%
   0       35.5      0.284     0.80
   1       35.5      0.353     0.99
   2       35.5     0.0989     0.28
   3       35.5      0.194     0.55
```

```
   *       142      0.929     0.65
-----------------------------------------------------------------------
@--- Callsites: 86 ---------------------------------------------------
-----------------------------------------------------------------------
 ID Lev File/Address        Line Parent_Funct           MPI_Call
  1   0 0x40a87f              main                  Request_free
  2   0 0x408bd7              main                  Cart_create
  3   0 0x40a731              main                  File_close
  4   0 0x409036              main                  File_close
  5   0 0x40a722              main                  File_write_all
  6   0 0x409027              main                  File_read_all
  7   0 0x408dcd              main                  Cart_shift
  8   0 0x408f55              main                  Type_commit
  9   0 0x40a85e              main                  Request_free
 10   0 0x40995f              main                  Send_init
 11   0 0x40a265              main                  Wait
 12   0 0x409d5e              main                  Recv_init
 13   0 0x40a83c              main                  Request_free
 14   0 0x409c3c              main                  Send_init
 15   0 0x408d9a              main                  Cart_rank
 16   0 0x408f1e              main                  Type_commit
 17   0 0x40a636              main                  Reduce
 18   0 0x40a025              main                  Start
 19   0 0x40a81b              main                  Request_free
-----------------------------------------------------------------------
@--- Aggregate Time (top twenty, descending, milliseconds) ----------
-----------------------------------------------------------------------
Call                Site       Time    App%    MPI%     COV
Allreduce             41        517    0.36   55.57    0.83
File_read_all          6       87.6    0.06    9.42    0.01
Wait                  11       75.9    0.05    8.17    0.34
File_open             49       70.5    0.05    7.59    0.17
File_write_all         5       56.3    0.04    6.06    0.00
File_close             3         49    0.03    5.27    1.99
Start                 77       14.9    0.01    1.61    0.09
Start                 24       12.6    0.01    1.35    0.01
Wait                  20         12    0.01    1.29    0.17
File_open             51       5.27    0.00    0.57    0.01
Reduce                30       3.98    0.00    0.43    1.92
Start                 59       2.55    0.00    0.27    0.67
Cart_create            2       1.94    0.00    0.21    0.23
Type_commit           36       1.84    0.00    0.20    0.14
Start                 47       1.78    0.00    0.19    0.19
File_close             4       1.62    0.00    0.17    0.44
Bcast                 22       1.59    0.00    0.17    0.17
File_set_view         39       1.34    0.00    0.14    0.03
Start                 75       1.34    0.00    0.14    0.33
Start                 58       1.15    0.00    0.12    0.36
-----------------------------------------------------------------------
@--- Aggregate Sent Message Size (top twenty, descending, bytes) ----------
```

```
-----------------------------------------------------------------------
Call              Site      Count     Total      Avrg  Sent%
Bcast               22          4  2.11e+03       528  84.62
Allreduce           41         72       288         4  11.54
Reduce              17          4        32         8   1.28
Reduce              29          4        32         8   1.28
Reduce              30          4        32         8   1.28
-----------------------------------------------------------------------
@--- Aggregate I/O Size (top twenty, descending, bytes) -----------------
-----------------------------------------------------------------------
Call              Site      Count     Total      Avrg   I/O%
File_read_all        6          4  1.45e+07  3.63e+06  50.00
File_write_all       5          4  1.45e+07  3.63e+06  50.00
-----------------------------------------------------------------------
@--- Callsite Time statistics (all, milliseconds): 212 ------------------
-----------------------------------------------------------------------
Name              Site Rank  Count      Max     Mean      Min   App%   MPI%
Allreduce           41    0     18     16.1     8.25     0.29   0.42  52.36
Allreduce           41    1     18       31     14.4    0.315   0.73  73.67
Allreduce           41    2     18    0.731   0.0563    0.015   0.00   1.02
Allreduce           41    3     18     7.41     5.95    0.019   0.30  55.20
Allreduce           41    *     72       31     7.17    0.015   0.36  55.57

Barrier             31    0      1     0.01     0.01     0.01   0.00   0.00
Barrier             31    1      1    0.148    0.148    0.148   0.00   0.04
Barrier             31    2      1    0.238    0.238    0.238   0.00   0.24
Barrier             31    3      1    0.185    0.185    0.185   0.00   0.10
Barrier             31    *      4    0.238    0.145     0.01   0.00   0.06

Bcast               22    0      1    0.315    0.315    0.315   0.00   0.11
Bcast               22    1      1    0.426    0.426    0.426   0.00   0.12
Bcast               22    2      1    0.376    0.376    0.376   0.00   0.38
Bcast               22    3      1    0.475    0.475    0.475   0.00   0.24
Bcast               22    *      4    0.475    0.398    0.315   0.00   0.17

Cart_coords         43    0      1     0.03     0.03     0.03   0.00   0.01
Cart_coords         43    1      1    0.057    0.057    0.057   0.00   0.02
Cart_coords         43    2      1    0.051    0.051    0.051   0.00   0.05
Cart_coords         43    3      1    0.053    0.053    0.053   0.00   0.03
Cart_coords         43    *      4    0.057   0.0478     0.03   0.00   0.02

Cart_create          2    0      1    0.635    0.635    0.635   0.00   0.22
Cart_create          2    1      1    0.396    0.396    0.396   0.00   0.11
Cart_create          2    2      1    0.507    0.507    0.507   0.00   0.51
Cart_create          2    3      1    0.406    0.406    0.406   0.00   0.21
Cart_create          2    *      4    0.635    0.486    0.396   0.00   0.21

Cart_rank           15    0      1    0.006    0.006    0.006   0.00   0.00
Cart_rank           15    *      1    0.006    0.006        0   0.00   0.00
```

Cart_rank	56	2	1	0.007	0.007	0.007	0.00	0.01
Cart_rank	56	*	1	0.007	0.007	0	0.00	0.00
Cart_rank	74	3	1	0.006	0.006	0.006	0.00	0.00
Cart_rank	74	*	1	0.006	0.006	0	0.00	0.00
Cart_rank	86	1	1	0.005	0.005	0.005	0.00	0.00
Cart_rank	86	*	1	0.005	0.005	0	0.00	0.00
Cart_shift	7	0	1	0.005	0.005	0.005	0.00	0.00
Cart_shift	7	1	1	0.005	0.005	0.005	0.00	0.00
Cart_shift	7	2	1	0.006	0.006	0.006	0.00	0.01
Cart_shift	7	3	1	0.005	0.005	0.005	0.00	0.00
Cart_shift	7	*	4	0.006	0.00525	0.005	0.00	0.00
Comm_free	45	0	1	0.011	0.011	0.011	0.00	0.00
Comm_free	45	1	1	0.007	0.007	0.007	0.00	0.00
Comm_free	45	2	1	0.008	0.008	0.008	0.00	0.01
Comm_free	45	3	1	0.007	0.007	0.007	0.00	0.00
Comm_free	45	*	4	0.011	0.00825	0.007	0.00	0.00
Dims_create	32	0	1	0.07	0.07	0.07	0.00	0.02
Dims_create	32	1	1	0.047	0.047	0.047	0.00	0.01
Dims_create	32	2	1	0.036	0.036	0.036	0.00	0.04
Dims_create	32	3	1	0.061	0.061	0.061	0.00	0.03
Dims_create	32	*	4	0.07	0.0535	0.036	0.00	0.02
File_close	3	0	1	48.8	48.8	48.8	0.14	17.20
File_close	3	1	1	0.055	0.055	0.055	0.00	0.02
File_close	3	2	1	0.057	0.057	0.057	0.00	0.06
File_close	3	3	1	0.067	0.067	0.067	0.00	0.03
File_close	3	*	4	48.8	12.2	0.055	0.03	5.27
File_open	49	0	1	15.2	15.2	15.2	0.04	5.37
File_open	49	1	1	21.2	21.2	21.2	0.06	6.02
File_open	49	2	1	15.2	15.2	15.2	0.04	15.41
File_open	49	3	1	18.8	18.8	18.8	0.05	9.70
File_open	49	*	4	21.2	17.6	15.2	0.05	7.59
File_read_all	6	0	1	22	22	22	0.06	7.75
File_read_all	6	1	1	21.6	21.6	21.6	0.06	6.14
File_read_all	6	2	1	21.9	21.9	21.9	0.06	22.14
File_read_all	6	3	1	22.1	22.1	22.1	0.06	11.36
File_read_all	6	*	4	22.1	21.9	21.6	0.06	9.42
File_set_view	34	0	1	0.08	0.08	0.08	0.00	0.03
File_set_view	34	1	1	0.08	0.08	0.08	0.00	0.02
File_set_view	34	2	1	0.078	0.078	0.078	0.00	0.08
File_set_view	34	3	1	0.079	0.079	0.079	0.00	0.04

```
File_set_view      34   *     4    0.08   0.0793    0.078   0.00   0.03


File_write_all      5   0     1    14.1    14.1    14.1   0.04   4.97
File_write_all      5   1     1    14.1    14.1    14.1   0.04   3.99
File_write_all      5   2     1    14.1    14.1    14.1   0.04  14.23
File_write_all      5   3     1    14.1    14.1    14.1   0.04   7.25
File_write_all      5   *     4    14.1    14.1    14.1   0.04   6.06

Recv_init          12   0     1   0.002   0.002   0.002   0.00   0.00
Recv_init          12   *     1   0.002   0.002       0   0.00   0.00

Recv_init          21   0     1   0.008   0.008   0.008   0.00   0.00
Recv_init          21   2     1   0.001   0.001   0.001   0.00   0.00
Recv_init          21   *     2   0.008  0.0045       0   0.00   0.00

Recv_init          27   0     1   0.001   0.001   0.001   0.00   0.00
Recv_init          27   1     1   0.001   0.001   0.001   0.00   0.00
Recv_init          27   *     2   0.001   0.001       0   0.00   0.00

Recv_init          52   0     1   0.001   0.001   0.001   0.00   0.00
Recv_init          52   1     1   0.001   0.001   0.001   0.00   0.00
Recv_init          52   *     2   0.001   0.001       0   0.00   0.00

Recv_init          54   2     1   0.001   0.001   0.001   0.00   0.00
Recv_init          54   *     1   0.001   0.001       0   0.00   0.00

Recv_init          60   2     1   0.002   0.002   0.002   0.00   0.00
Recv_init          60   *     1   0.002   0.002       0   0.00   0.00

Recv_init          62   2     1   0.011   0.011   0.011   0.00   0.01
Recv_init          62   3     1   0.002   0.002   0.002   0.00   0.00
Recv_init          62   *     2   0.011  0.0065       0   0.00   0.00

Reduce             17   0     1   0.002   0.002   0.002   0.00   0.00
Reduce             17   1     1   0.002   0.002   0.002   0.00   0.00
Reduce             17   2     1   0.002   0.002   0.002   0.00   0.00
Reduce             17   3     1   0.002   0.002   0.002   0.00   0.00
Reduce             17   *     4   0.002   0.002   0.002   0.00   0.00


Request_free        1   0     3   0.003  0.00233   0.002   0.00   0.00
Request_free        1   1     3   0.002  0.00133   0.001   0.00   0.00
Request_free        1   2     3   0.002  0.00133   0.001   0.00   0.00
Request_free        1   3     3   0.002   0.002   0.002   0.00   0.00
Request_free        1   *    12   0.003  0.00175   0.001   0.00   0.00


Send_init          10   0     1   0.002   0.002   0.002   0.00   0.00
Send_init          10   2     1   0.001   0.001   0.001   0.00   0.00
```

```
Send_init        10    *     2    0.002   0.0015       0   0.00   0.00

Send_init        14    0     1    0.001    0.001    0.001   0.00   0.00
Send_init        14    1     1    0.001    0.001    0.001   0.00   0.00
Send_init        14    *     2    0.001    0.001        0   0.00   0.00

Send_init        38    0     1    0.001    0.001    0.001   0.00   0.00
Send_init        38    *     1    0.001    0.001        0   0.00   0.00

Send_init        48    0     1    0.001    0.001    0.001   0.00   0.00
Send_init        48    1     1    0.117    0.117    0.117   0.00   0.03
Send_init        48    *     2    0.117    0.059        0   0.00   0.01

Send_init        57    2     1    0.001    0.001    0.001   0.00   0.00
Send_init        57    3     1    0.001    0.001    0.001   0.00   0.00
Send_init        57    *     2    0.001    0.001        0   0.00   0.00

Send_init        61    2     1    0.002    0.002    0.002   0.00   0.00
Send_init        61    *     1    0.002    0.002        0   0.00   0.00


--------------------------------------------------------------------------
@--- Callsite Message Sent statistics (all, sent bytes) --------------------
--------------------------------------------------------------------------
Name             Site Rank  Count      Max      Mean       Min       Sum
Allreduce          41    0     18        4         4         4        72
Allreduce          41    1     18        4         4         4        72
Allreduce          41    2     18        4         4         4        72
Allreduce          41    3     18        4         4         4        72
Allreduce          41    *     72        4         4         4       288

Bcast              22    0      1      528       528       528       528
Bcast              22    1      1      528       528       528       528
Bcast              22    2      1      528       528       528       528
Bcast              22    3      1      528       528       528       528
Bcast              22    *      4      528       528       528      2112

Reduce             17    0      1        8         8         8         8
Reduce             17    1      1        8         8         8         8
Reduce             17    2      1        8         8         8         8
Reduce             17    3      1        8         8         8         8
Reduce             17    *      4        8         8         8        32

--------------------------------------------------------------------------
@--- Callsite I/O statistics (all, I/O bytes) ----------------------------
--------------------------------------------------------------------------
Name             Site Rank  Count      Max      Mean       Min       Sum
File_read_all       6    0      1 3.629e+06 3.629e+06 3.629e+06 3.629e+06
File_read_all       6    1      1 3.629e+06 3.629e+06 3.629e+06 3.629e+06
File_read_all       6    2      1 3.629e+06 3.629e+06 3.629e+06 3.629e+06
```

```
File_read_all       6    3       1 3.629e+06 3.629e+06 3.629e+06 3.629e+06
File_read_all       6    *       4 3.629e+06 3.629e+06 3.629e+06 1.452e+07

File_write_all      5    0       1 3.629e+06 3.629e+06 3.629e+06 3.629e+06
File_write_all      5    1       1 3.629e+06 3.629e+06 3.629e+06 3.629e+06
File_write_all      5    2       1 3.629e+06 3.629e+06 3.629e+06 3.629e+06
File_write_all      5    3       1 3.629e+06 3.629e+06 3.629e+06 3.629e+06
File_write_all      5    *       4 3.629e+06 3.629e+06 3.629e+06 1.452e+07

-----------------------------------------------------------------------
@--- End of Report ----------------------------------------------------
-----------------------------------------------------------------------
```

3.3. Βασικές έννοιες αποδοτικότητας σε παράλληλους υπολογιστές.

Στην ενότητα αυτή δίνουμε κάποιες βασικές έννοιες αποδοτικότητας παράλληλων προγραμμάτων για να γίνουν κατανοητά τα διαγράμματα απόδοσης στην στην ενότητα 3.4.

Η δείκτες που χρησιμοποιήσαμε για να μετρήσουμε την απόδοση των αποτελεσμάτων στην παράλληλη μηχανή ARIS, είναι ο λόγος επιτάχυνσης S(N) (speed up), και η αποδοτικότητα (Efficiency) E(N) .

Speed Up

Είναι η αναλογία του χρόνου εκτέλεσης ενός προγράμματος σε έναν επεξεργαστή με το χρόνο εκτέλεσης σε N επεξεργαστές.

$$S(n) = \frac{t_s}{t_n}$$

- Όπου t_s είναι ο σειριακός χρόνος εκτέλεσης και t_n ο παράλληλος χρόνος εκτέλεσης.

- Το βέλτιστο speedup σε N επεξεργαστές είναι: S(N)=N
  Όπου N είναι ο αριθμός των επεξεργαστών.

Efficiency (Αποδοτικότητα)

Με τον όρο αποδοτικότητα ορίζουμε το μέτρο της αποτελεσματικότητας της χρήσης του επεξεργαστή

$$E = \frac{S(n)}{n}$$

- Η μεγαλύτερη τιμή που μπορεί να πάρει το E(n) είναι 1 οπού έχουμε γραμμική επιτάχυνση και γενικότερα όταν E=1 το πρόγραμμα μας είναι αποδοτικό.

Πηγές που μειώνουν την αποδοτικότητα (Sources of overhead)

- Περίοδοι κατά τις οποίες οι επεξεργαστές είναι σε αδράνεια
- Επιπλέον υπολογισμούς που απαιτούνται για την παράλληλη εκτέλεση
- Ο χρόνος επικοινωνίας για την αποστολή μηνυμάτων μεταξύ των διεργασιών που εκτελούνται παράλληλα

Cost (Κόστος)

‘Cost’ ή αλλιώς ‘processor-time product’ είναι η τιμή του κόστους χρήσης επιπλέον επεξεργαστών.

cost = (execution time) χ (number of processors)

$$Cost = \frac{t_s n}{S(n)} = \frac{t_s}{E}$$

Παράδειγμα:

Έχουμε ένα πρόβλημα που χρειάζεται 10 λεπτά για να τρέξει σε έναν επεξεργαστή. Στην παράλληλη εκτέλεση τρέχει σε 4 επεξεργαστές και χρειάζεται 4 λεπτά ποια θα είναι η τιμή τον S(n),E(n),Cost ?

S(4)= 10/4 = 2.5

E(4)= 2.5/4= 0.625 = 62.5%

Cost = 10/0.625 = 16

Έχουμε έναν καινούριο αλγόριθμο για το ίδιο πρόβλημα που τρέχει σε 16 λεπτά σε έναν επεξεργαστή και στην παράλληλη εκτέλεση χρειάζεται 4 λεπτά σε 4 επεξεργαστές.

S(4)= 16/4 = 4 (γραμμική επιτάχυνση)

E(4)= 4/4 = 100 % (μέγιστη απόδοση)

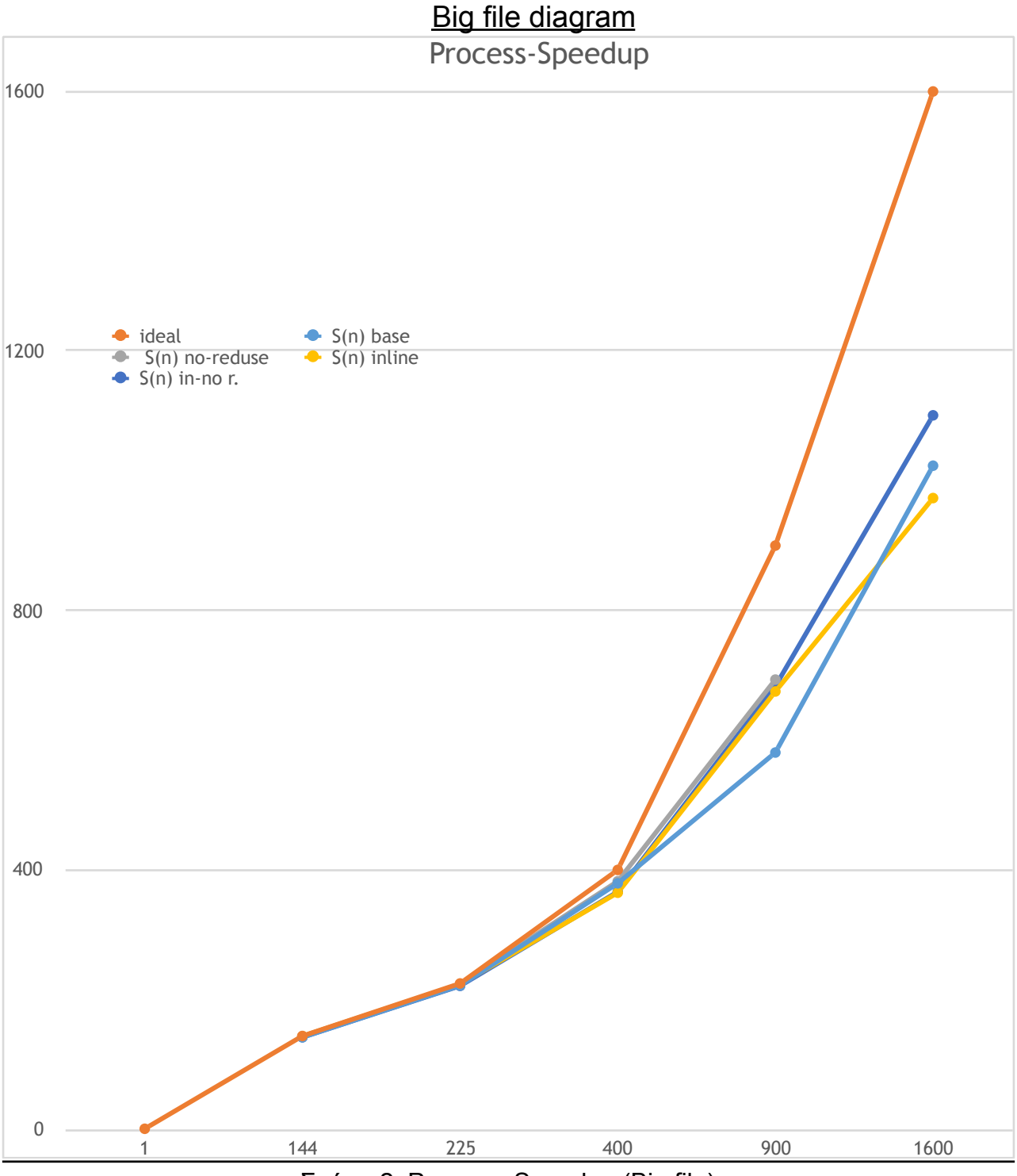
Cost = 16/1 = 16

Scalability (Κλιμάκωση)

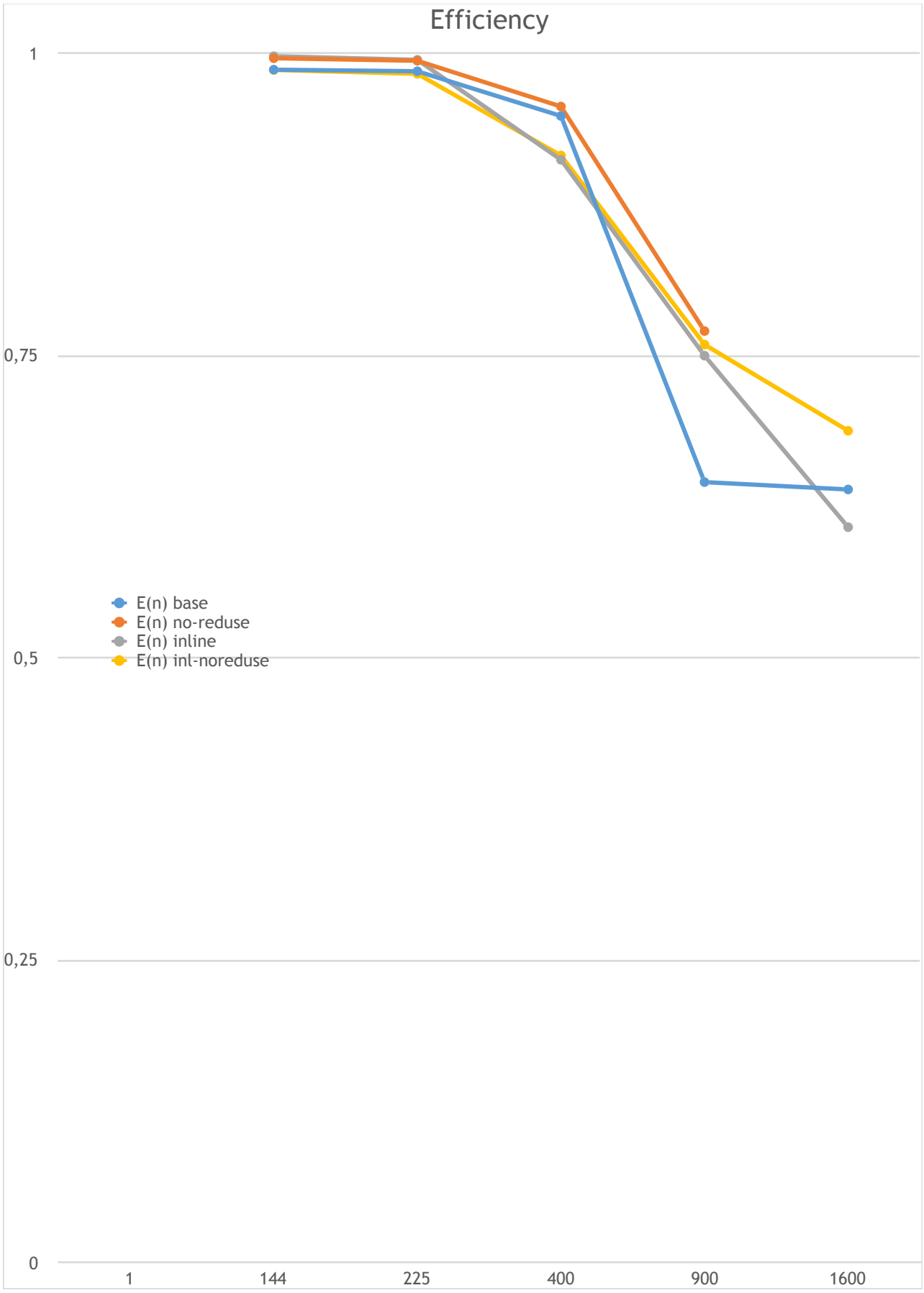
Με τον όρο Scalability αναφερόμαστε στη δυνατότητα ενός παράλληλου αλγορίθμου με μια τιμή αποδοτικότητας E σε p επεξεργαστές να λύνει ένα μεγαλύτερο πρόβλημα με την ίδια αποδοτικότητα σε p+1 επεξεργαστές.

3.4. Διαγράμματα απόδοσης του προγράμματός μας

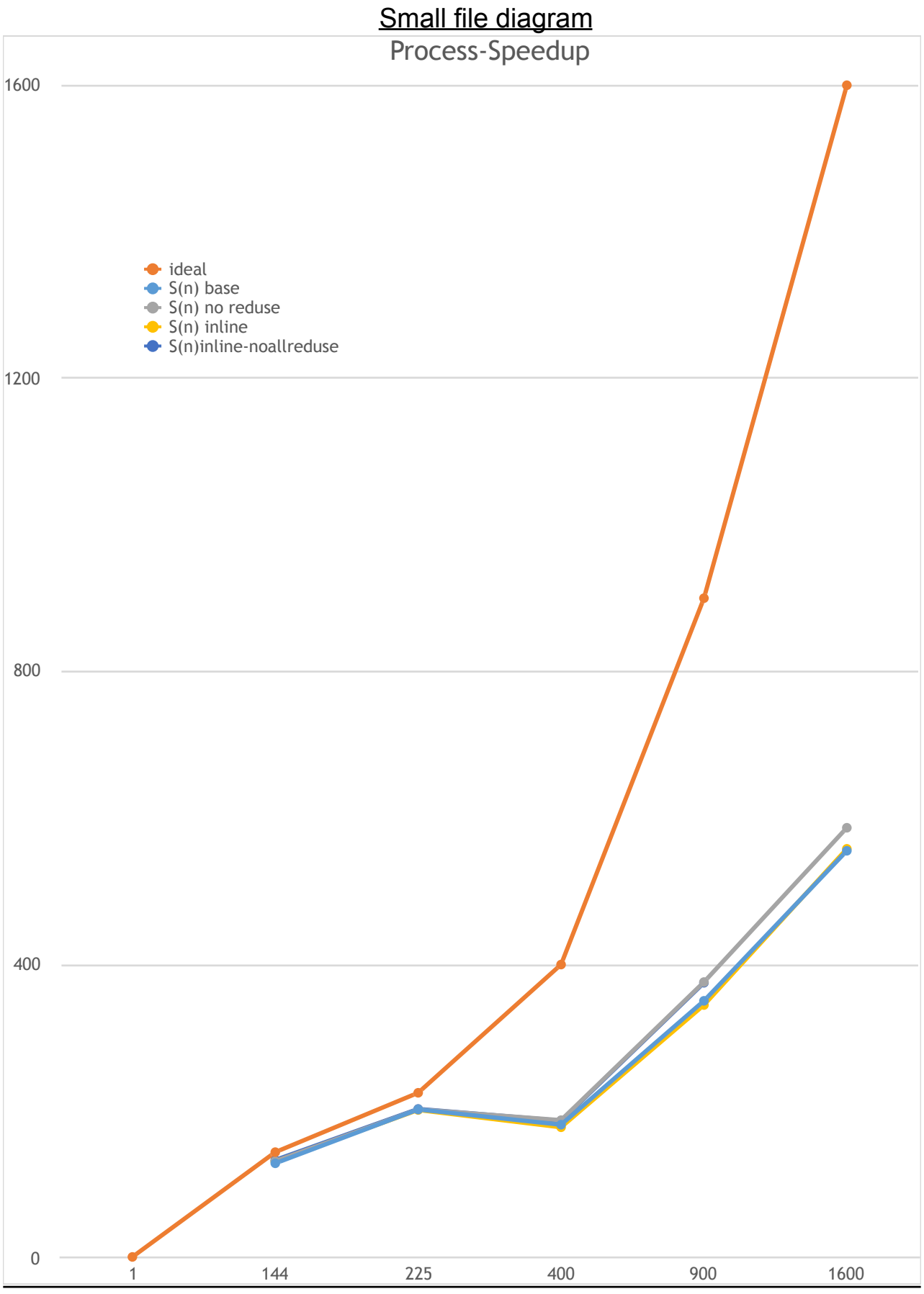
Σε αυτή την ενότητα παρουσιάζουμε τα διαγράμματα απόδοσης όλων των εκδόσεων του προγράμματός μας. Το base είναι το αρχικό πρόγραμμα, το no-reduce είναι με την αφαίρεση της συνάρτησης ALLReduce (γραμμή 471 στο κώδικα), το inline είναι με τον ορισμό inline συναρτήσεων και το inline-noreduce ο συνδυασμός των δύο παραπάνω. Δίνουμε τις γραφικές παραστάσεις εκτελέσεων για μια μεγάλη εικόνα 700mb και για μια μικρή 14mb.



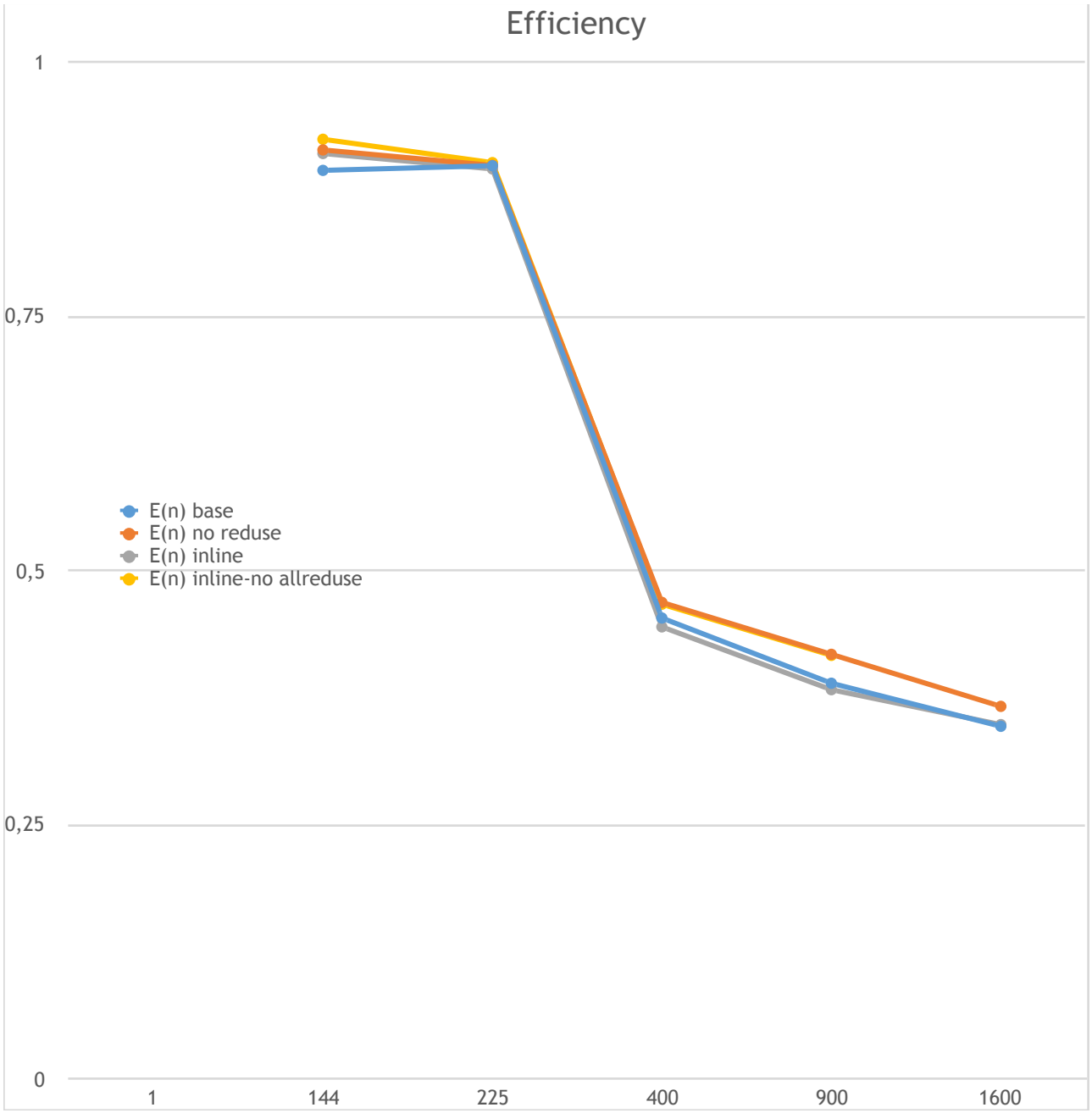
Σχήμα 2: Process_Speedup (Big file)



Σχήμα 3: Eficiency (Big file)



Σχήμα 4: Process-Speedup (Small file)



Σχήμα 5: Efficiency (Small file)

Συμπεράσματα

Από τις γραφικές παραστάσεις παρατηρούμε ότι η τελευταία έκδοση του προγράμματός μας παρουσιάζει έστω και σε μικρό βαθμό το καλύτερο efficiency και για small data (μικρότερο πρόβλημα) και για big data (μεγαλύτερο πρόβλημα). Συμπεραίνουμε λοιπόν ότι η τελευταία έκδοση κλιμακώνει καλύτερα από τις υπόλοιπες.(Scalability)

ΠΙΝΑΚΑΣ ΟΡΟΛΟΓΙΑΣ

Ξενόγλωσσος όρος	**Ελληνικός Όρος**
Avail	Όφελος
Benchmark	Ένα πρότυπο ή σημείο αναφοράς το οποίο αποτελεί μέτρο σύγκρισης
Cartesian Topology	Καρτεσιανή Τοπολογία
Cluster	Σύμπλεγμα
Command	Εντολή
Compiler	Μεταγλωττιστής
Computer nodes	Υπολογιστικές Μονάδες
Dependency	Εξάρτηση
Documentation	Υλικό που παρέχει επίσημες πληροφορίες για ένα προϊόν.
Efficiency	Αποδοτικότητα
Environment Variables	Μεταβλητές περιβάλλοντος
Errors	Σφάλματα
File System	Σύστημα αρχείων
Hardware Control	Έλεγχος υλικού
High Performance Computer	Υπολογιστές Υψηλής Απόδοσης
Library	Βιβλιοθήκη
Load/Unload	Φόρτωση / Εκφόρτωση
Monitoring	Παρακολούθηση
Multi-threaded	Πολυνηματικός
Open-source	Ελεύθερο Λογισμικό
Operating System	Λειτουργικό Σύστημα
Partition	Διχοτόμιση
Pending Jobs	Εργασίες Εν Αναμονή
Priority	Προτεραιότητα
Private key	Ιδιωτικό Κλειδί
Process	Διεργασία
Public Key	Δημόσιο Κλειδί
Purge	Καθαρίζω

Queue	Ουρά
Quota	Ποσοστό
Reservation	Κράτηση
Resources	Πόροι
Scalability	Κλιμάκωση
Software Environment	Περιβάλλον λογισμικού
Speed Up	Επιτάχυνση
Supercomputer	Υπερυπολογιστής
Switch	Μεταγωγέας
System Failure	Σφάλμα Συστήματος
System Management	Σύστημα διαχείρισης
Topology	Τοπολογία
Usage	Χρήση
Workload Manager	Διαχειριστής Φόρτου Εργασίας

ΣΥΝΤΜΗΣΕΙΣ – ΑΡΚΤΙΚΟΛΕΞΑ – ΑΚΡΩΝΥΜΙΑ

ABYSS	Assembly By Short Sequences
ARIS	Advanced Research Information System
CPU	Central Processing Unit
GPFS	General Parallel File System
HPC	High-performance computing
ID	Identification Data
Infiniband FDR	Fourteen Data Rate
Keygen	Key Generation
MPI	Message Passing Interface
Man Page	Manual Page
OPENMP	Open Multi-Processing
OS	Operating System
PCI Bus	Peripheral Component Interconnect Bus
SSH	Secure Shell
SLURM	Simple Linux Utility for Resource Management
TFLOPS	Trillion Floating Points
mpiP	MPI Profiling
rgb	Red Green Blue
xCAT	Extreme Cloud Administration Toolkit
ΕΔΕΤ	Ελληνικό Δίκτυο Έρευνας και Τεχνολογίας

ΠΑΡΑΡΤΗΜΑ

(Αναφορά του Project μας στον ARIS)

Με το πέρας της προθεσμίας της πρόσβασης στον ARIS και την ολοκλήρωση του project ζητείται από το προσωπικό μια αναφορά με τα αποτελέσματα.Σας παραθέτουμε την αναφορά του δικού μας project.

hpc.grnet.gr Final Report Form – Preparatory Call (Types A and B)

MONTH YEAR round of evaluation

1. General information (for Type A and Type B projects)

1.1. Proposal ID

1.2. Type of preparatory proposal granted

B – Code Development

1.3. Period of access to the ARIS

01/11/2015 - 01/05/2016

2. Project information

2.1. Project name and description to which corresponds the developed and optimized code

ParSysStudy : Extended Study of Student Projects in Parallel Systems

2.2. Research field

Mathematics and Computer Science

2.3. Institutions and research team members

National and Kapodistrian University of Athens
Department of Informatics and Telecommunications

Team members:
Prof Yiannis Cotronis
Student George Nafpaktitis
Student Leonidas Dimakis

2.4. Summary of the project interest

Notice that should be mentioned the scientific cases and goals related to the project as well as technical goals (performance, parallel scalability, etc.)

If it is suitable with the precedent comments, fill in the field with the same text used in the application form.

The main interest of the project was the observation of how a program scales according to the size of data which are given for processing and the number of cores that are allocated for parallel process.The goal was to accomplish better scaling by improving the code of the program and the algorithm.

3. Scalability testing (for Type A and Type B projects)

3.1. Summary of the obtained results from the scalability testing

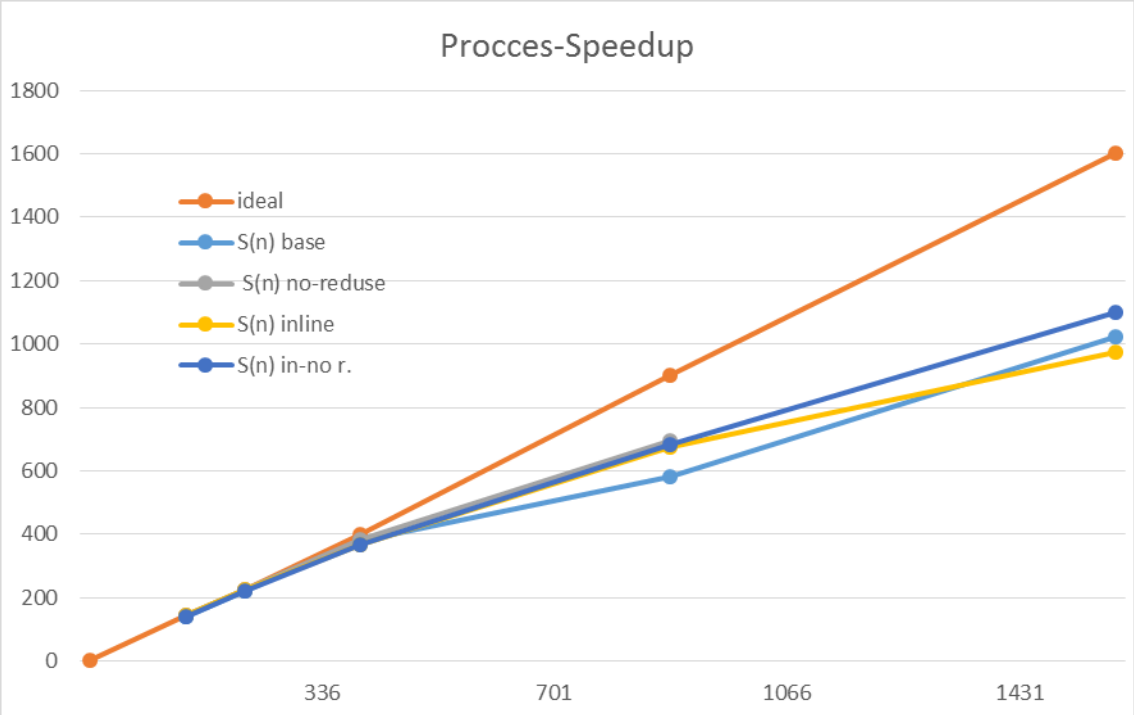
Show the scaling behavior of your application. Which progress did you achieve? Does it fulfill your expectations? If not, what were the reasons?

On the two diagrams below we show the scalability of our program.One diagram is for big data and one for small data. There are 4 versions of our program each one with some optimisations. As it seems the fourth version has a better scalability in big data and it is closer to ideal.

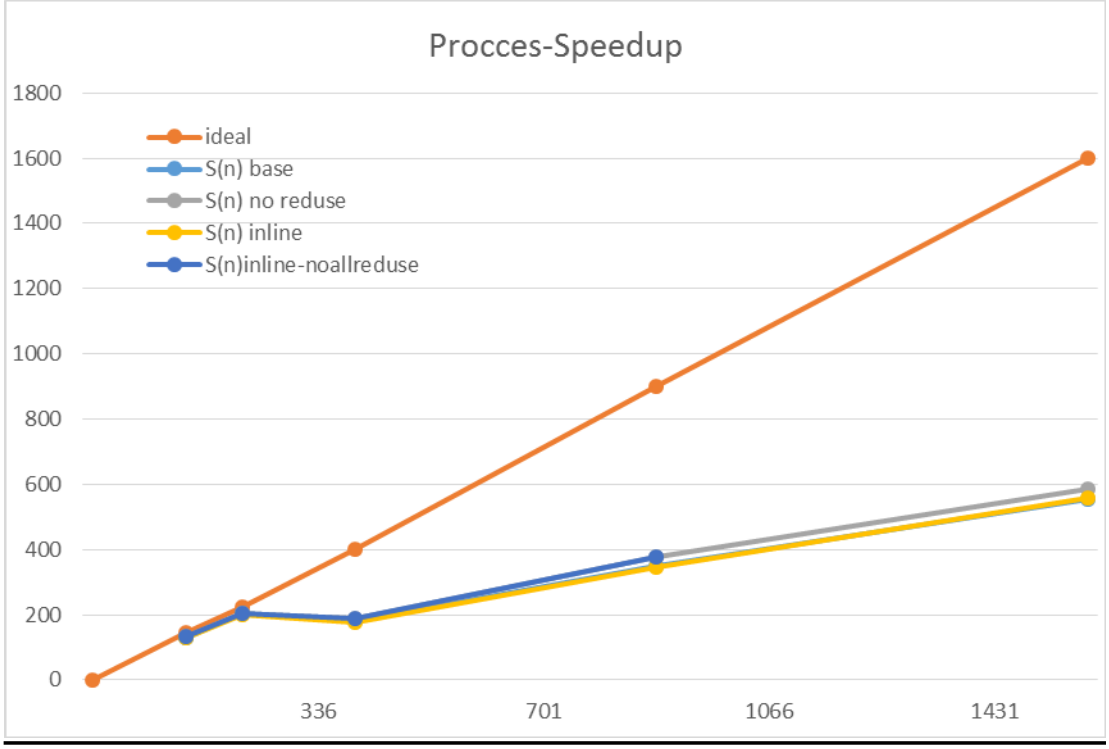
3.2. Images or graphics showing results from the scalability testing *(Minimum resolution of 300 dpi)*

Please attach the images to this form.
All tables and figures (including photographs, schemas, graphs and diagrams) should be numbered with Arabic numerals (1,2,...n) and include a descriptive caption.

Big Data



Small Data



3.3. Data to deploy scalability curves

A) Some typical user test cases
Please include the data for each test case.

Big Data Version 1

Number of cores	Wall clock time (seconds)	Speed-up vs the first one	Number of Nodes	Number of process
144	475	142	8	18
225	304	221	15	15
400	178	379	20	20
900	116	580	45	20
1600	66	1023	80	20

Big Data Version 2

Number of cores	Wall clock time (seconds)	Speed-up vs the first one	Number of Nodes	Number of process
144	487	143	8	18
225	312	223	15	15
400	183	382	20	20
900	100	693	45	20
1600	Time out		80	20

Big Data Version 3

Number of cores	Wall clock time (seconds)	Speed-up vs the first one	Number of Nodes	Number of process
144	474	143	8	18
225	304	223	15	15
400	186	364	20	20
900	101	675	45	20
1600	70	973	80	20

Big Data Version 4

Number of cores	Wall clock time (seconds)	Speed-up vs the first one	Number of Nodes	Number of process
144	487	142	8	18
225	312	221	15	15
400	188	366	20	20
900	101	683	45	20
1600	62	1100	80	20

Small Data Version 1

Number of cores	Wall clock time (seconds)	Speed-up vs the first one	Number of Nodes	Number of process
144	11	128	8	18
225	7	202	15	15
400	8	181	20	20
900	4	350	45	20
1600	2	555	80	20

Small Data Version 2

Number of cores	Wall clock time (seconds)	Speed-up vs the first one	Number of Nodes	Number of process
144	11	131	8	18
225	7	202	15	15
400	7	187	20	20
900	3	376	45	20
1600	2	586	80	20

Small Data Version 3

Number of cores	Wall clock time (seconds)	Speed-up vs the first one	Number of Nodes	Number of process
144	11	131	8	18
225	7	201	15	15
400	8	177	20	20
900	4	344	45	20
1600	2	557	80	20

Small Data Version 4

Number of cores	Wall clock time (seconds)	Speed-up vs the first one	Number of Nodes	Number of process
144	11	133	8	18
225	7	202	15	15
400	8	186	20	20
900	4	375	45	20

3.4. Publications or reports regarding the scalability testing. *(Format: Author(s). "Title". Publication, volume, issue, page, month year)*

4. Development and optimization (for Type B - developments projects)

4.1. General description of the work done in the project *(unlimited number of words)*
Please, mention the technical and algorithmic methods and programming techniques employed, the use of profiling tools when applicable and the use of numerical libraries when applicable.

The main goal of our project was to optimise the performance of our program by improving the code and the algorithm. Our program processes images according to a given filter. The image is processed by this filter with many iterations and that's why the execution requires significant amounts of resources.Our program uses MPI with persistent communication and virtual cartesian topology.It splits the image in several sub-images according to given resources and assigns every sub-image to a separate process for processing separately.

4.2. Summary of the obtained results from the enabling process *(Maximum 500 words)*

Please describe the effort you spent. Which progress did you achieve? Please describe in detail which enabling work was performed (porting, work on algorithms, I/O…etc.). Which problems did you experience?

We achieve to optimise the scalability of our program compared with the size of the problem and the amount of the using processors.You can see that at the diagrams we attached on sector 3. The changes we made where mainly in unnecessary loops our algorithm was doing and optimisations on code.Some of the optimisations had to do with MPI communication.

4.2. If applicable, which tools did you use to analyze your code? (e.g. Scalasca, Vampir…etc.) *(Maximum 500 words)*

MpiP was the tool that we used the most.With this way we found the bottlenecks and tried to resolve them.

4.3. What are the main actions that you did for optimization or improvement of your code on ARIS? What feature was to be optimized? What was the bottleneck? What solution did you use (if any)? *(Maximum 500 words)*

Thanks to MpiP we found that an MPI function we called many times caused a lot of bottleneck and we tried to reduce the calls.Furthermore we found the call of the main function which does all the calculations for the image process caused a lot of bottleneck.To reduce the delay we added the inline feature to this function.

4.4. Publications or reports regarding the development and optimization. *(Format: Author(s). "Title". Publication, volume, issue, page, month year)*

5. Main results (for Type A and Type B projects)

What are your conclusions? What do you think of the usability of ARIS? Which is the relevance of the obtained results for the stated scientific goals? Please, explain the outlook on the possible future work.

From our experience with ARIS we think that is a very powerful supercomputer that can withstand the requirements of the greek research and facilitate scientist's work.But without good information and knowledge of the users it can be easily loose its value with bad use.For example one thing we observed was about the usage dissimilarity along 24hours of the day.We saw that during the day all resources was reserved(with result big pending queues) when during night very few there was used.This is obviously a problem of time distribution which could be resolved by more well-informed users and better management.

6. Feedback and technical deployment

6.1. Feedback on the ARIS *(Maximum 500 words)*

6.2. Explanation of how the computer time was used compared with the work plan presented in the proposal. Justification of discrepancies, especially if the computer time was not completely used. *(Maximum 500 words)*

6.3. Please, let us know if you plan to apply for a regular hpc.grnet.gr project? If not, explain us why. *(Maximum 500 words)*

ΑΝΑΦΟΡΕΣ

1. http://doc.aris.grnet.gr
2. https://hpc.grnet.gr
3. An Introduction to Parallel Programming, 1st Edition By Peter Pacheco ISBN: 9780123742605
4. Parallel Programming with MPI By Peter Pacheco ISBN: 9781558603394