Εθνική υπερυπολογιστική υποδομή ARIS

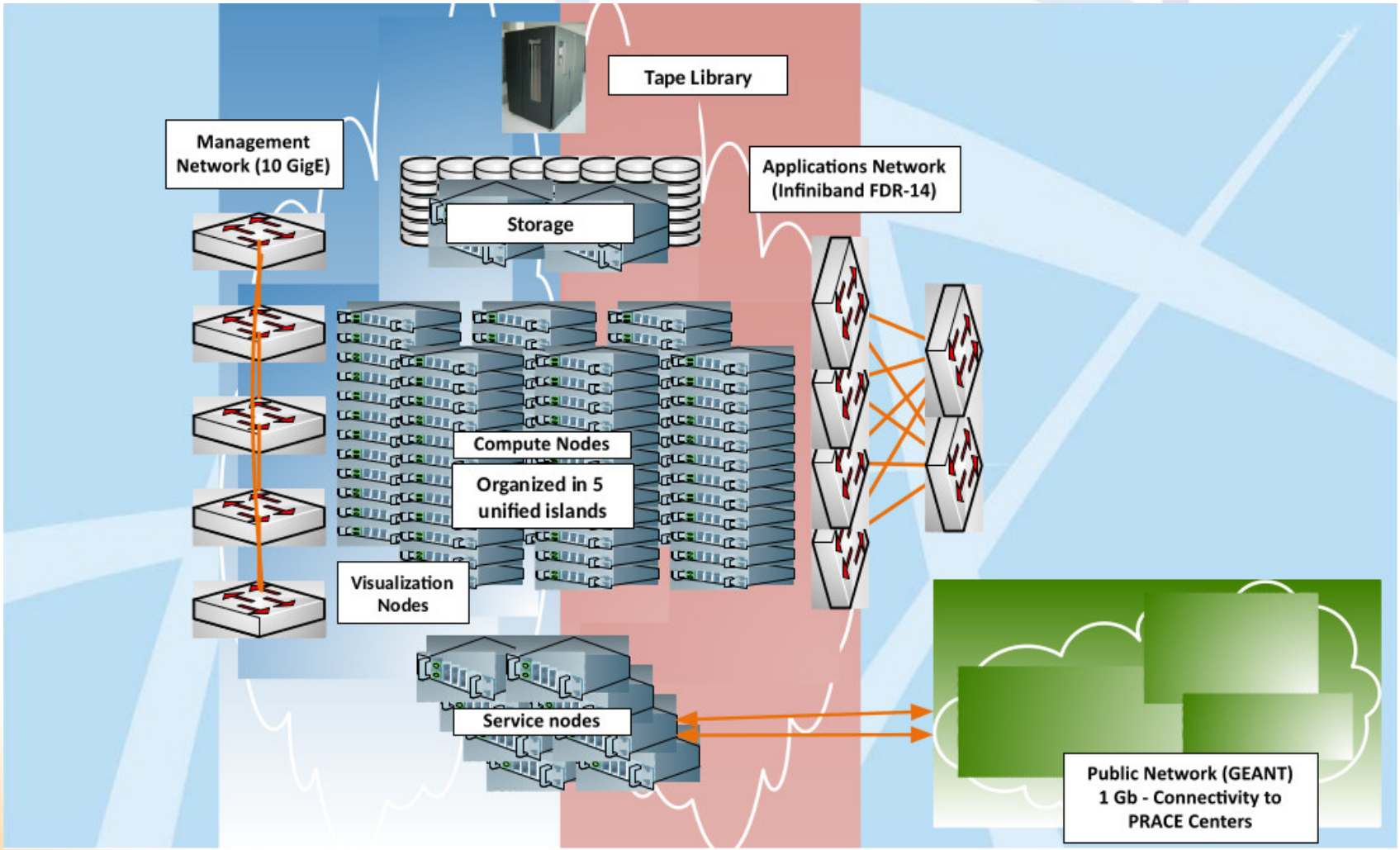
Δρ. Δημήτρης Ντελλής
ntell [at] grnet.gr

GRNET

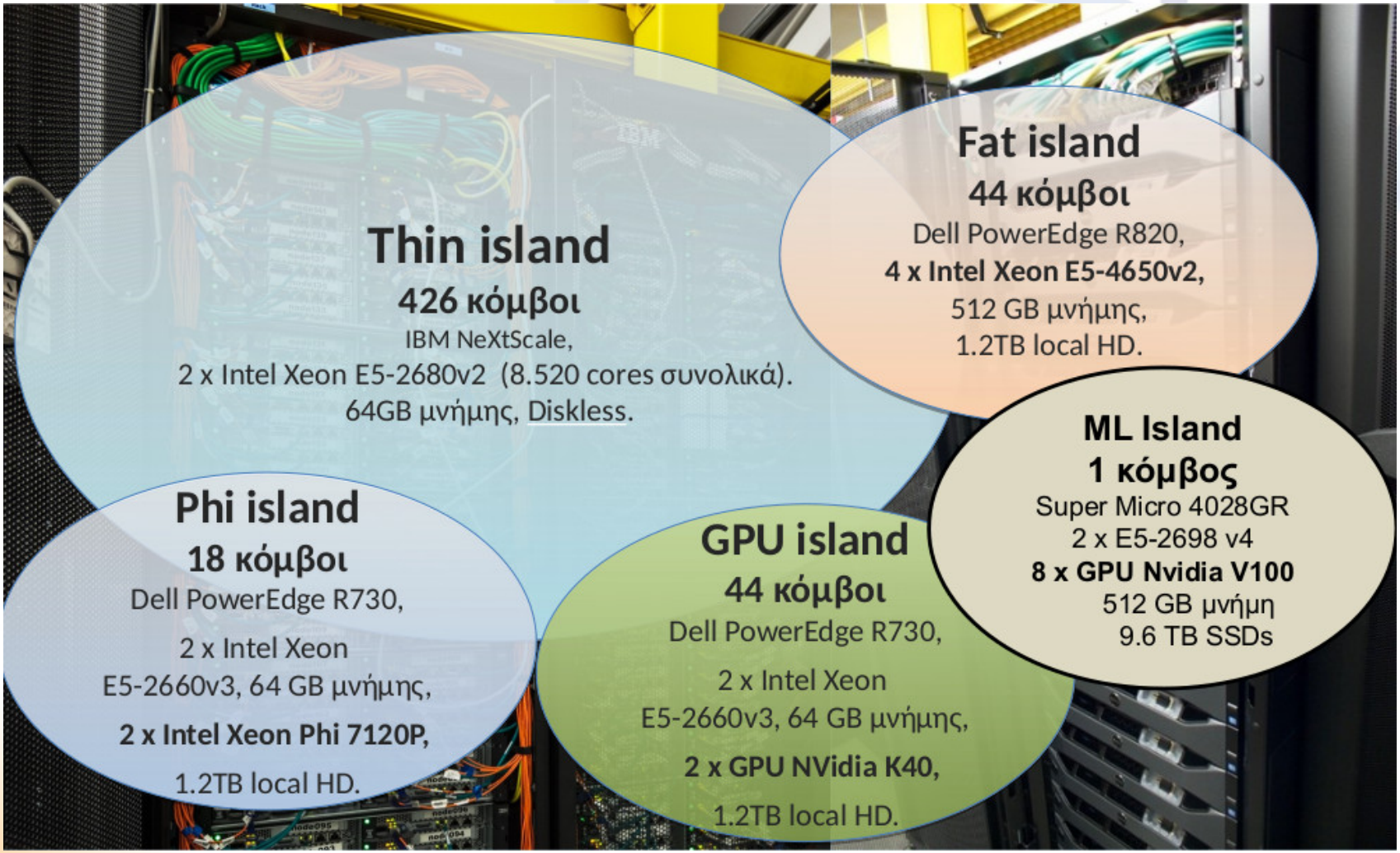
Περιεχόμενα I

- Περιγραφή Συστήματος
- Χρήση Συστήματος
- Environment Modules - Software
- Resources Manager / Batch System
- Software
- Καλές - Κακές πρακτικές σε DL

Περιγραφή Συστήματος



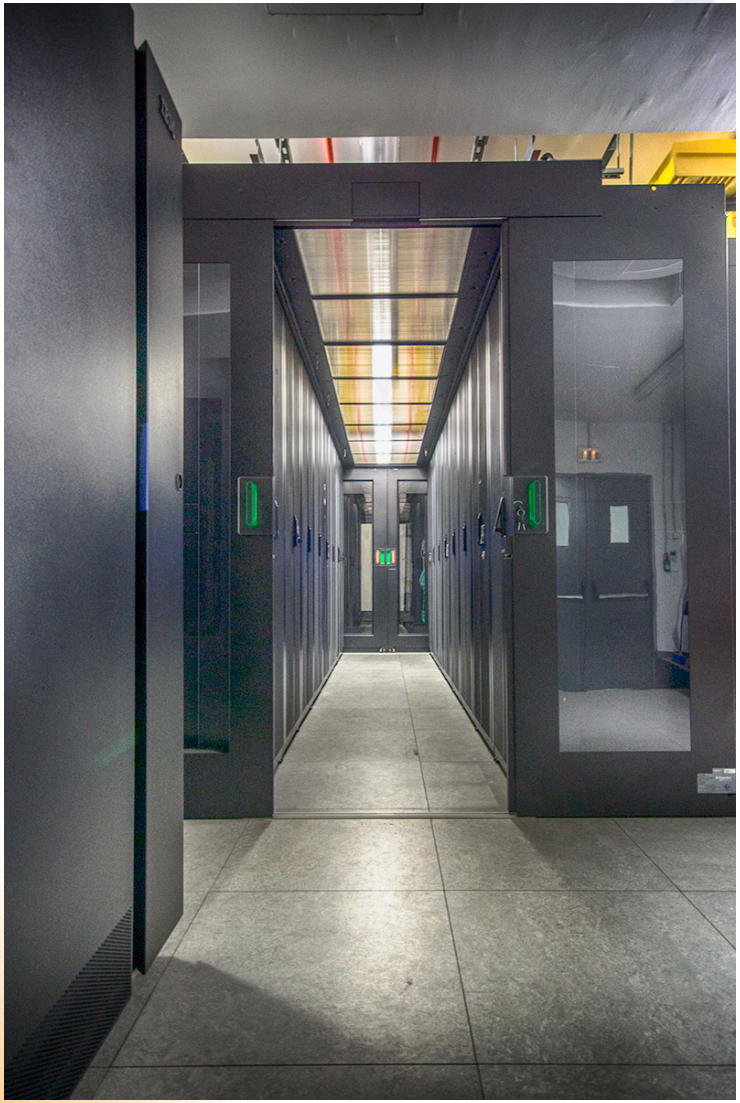
Περιγραφή Συστήματος



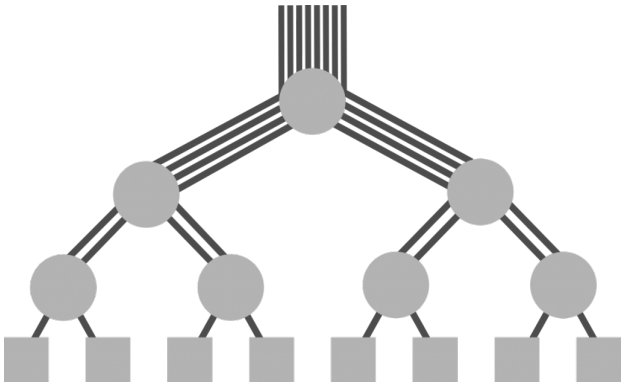
Περιγραφή Συστήματος

- **Χώρος εγκατάστασης: Κτίριο Υπουργείου Παιδείας Ερευνας και Θρησκευμάτων, Μαρούσι**
- **Δύο φάσεις ανάπτυξης:**
  - Α' Φάση: Στόχος εφαρμογές υψηλής παραλληλίας
  - Β' Φάση: Εφαρμογές με μεγάλες απαιτήσεις σε μνήμη και εκμετάλλευση τεχνολογιών συνεπεξεργαστών/επιταχυντών

Περιγραφή Συστήματος



Περιγραφή Συστήματος



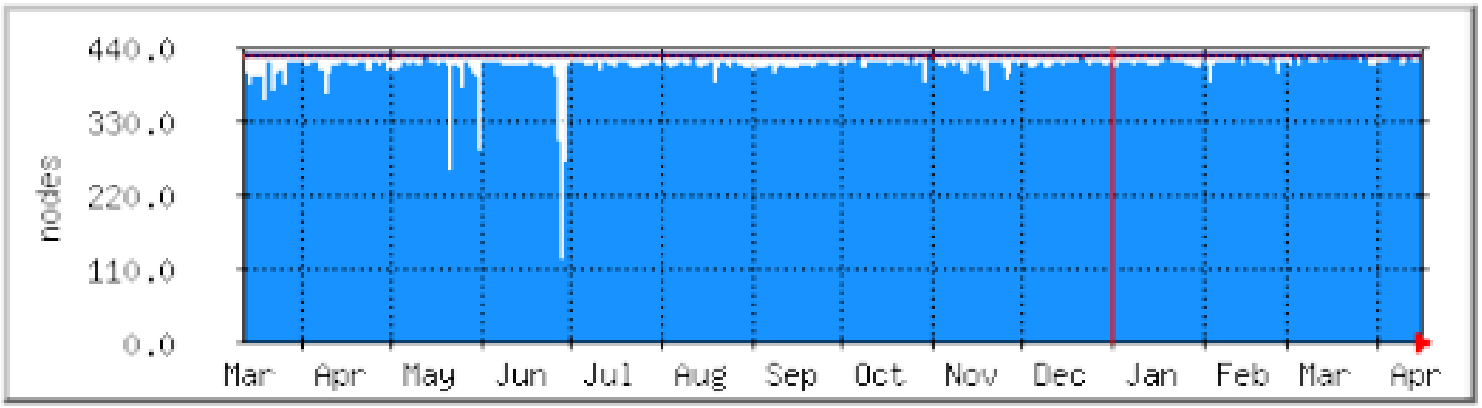
Περιγραφή Συστήματος

Πρόσβαση στο Σύστημα

- login nodes : Προσβάσιμοι μέσω Internet, MONO μέσω SSH, από δηλωμένες IPs/Δίκτυα.
- Σύνδεση SSH με χρήση κλειδιού **MONO**
- Ιδρυματικά VPN pools ενεργοποιημένα
- compute nodes : χρησιμοποιούνται μόνο από τα jobs, δέν είναι άμμεσα προσβάσιμα, δεν έχουν πρόσβαση internet.
- Οι SSH συνδέσεις από το ARIS προς οπουδήποτε **ΔΕΝ** επιτρέπονται.
- Μεταφορές αρχείων προς τα ιδρύματα : Αντί *ARIS* $\longrightarrow$ *PC* *put*, *PC* $\longrightarrow$ *ARIS* *get*.
- File system : GPFS, 4 partitions, 2 PB raw capacity

Στατιστικά χρήσης

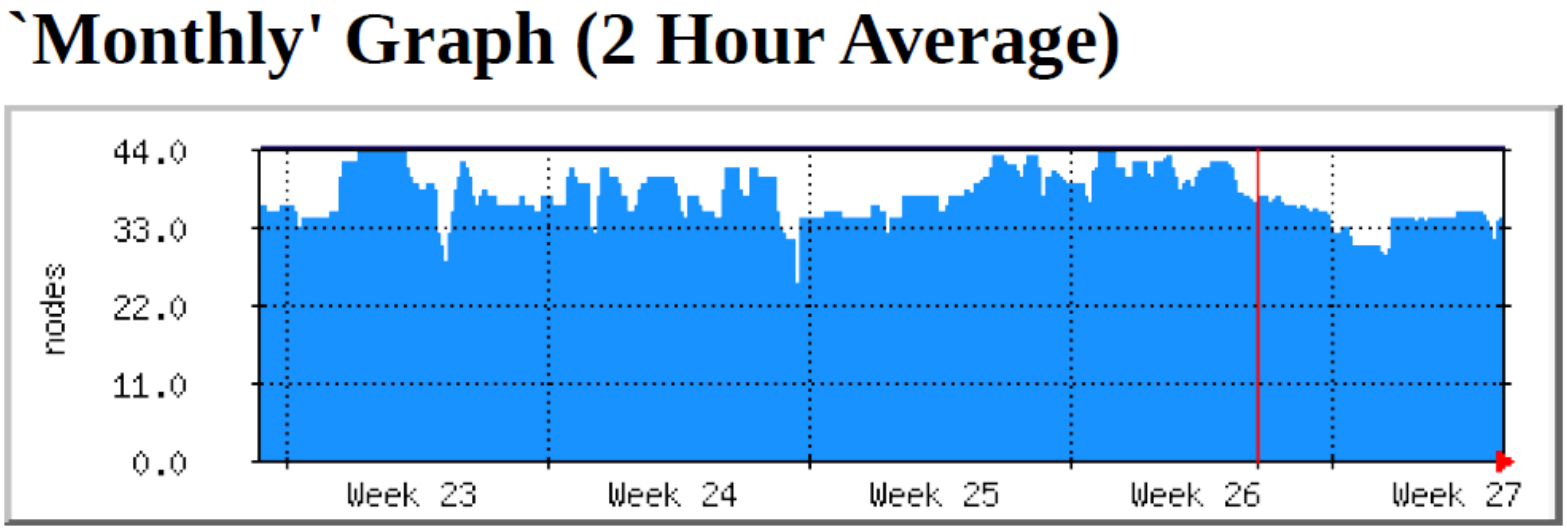
Thin Nodes



	Max	Average	Current
Nodes Allocated	421 Nodes (98.8%)	404 Nodes (94.8%)	420 Nodes (98.6%)
Total Nodes	426 Nodes (100.0%)	426 Nodes (100.0%)	426 Nodes (100.0%)

Στατιστικά χρήσης

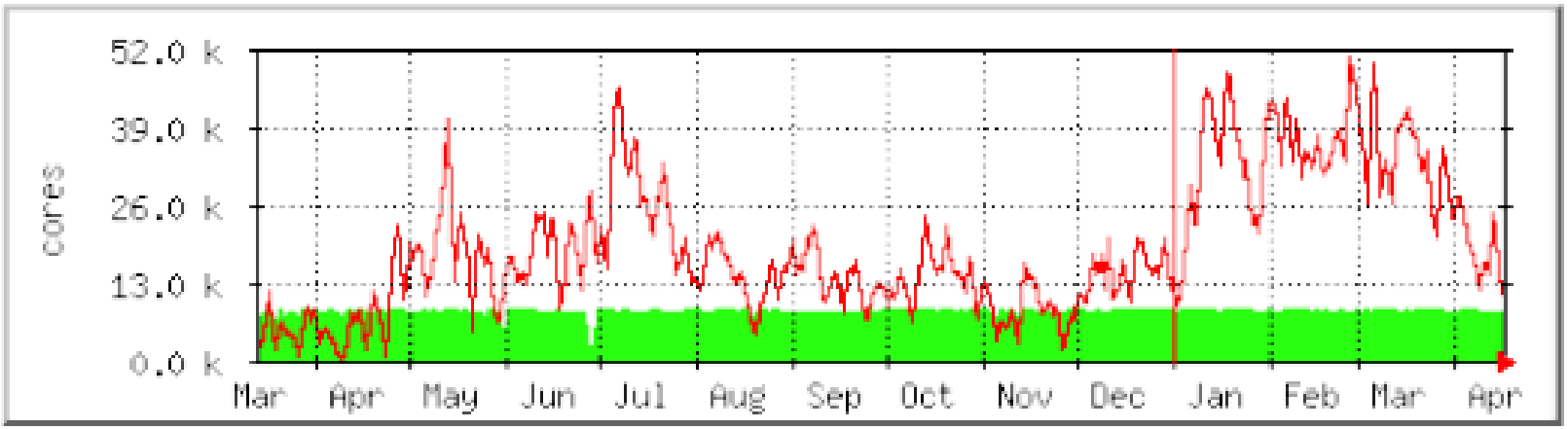
GPU Nodes



	Max	Average	Current
Nodes Allocated	44 Nodes (100.0%)	37 Nodes (84.1%)	34 Nodes (77.3%)
Total Nodes	44 Nodes (100.0%)	44 Nodes (100.0%)	44 Nodes (100.0%)

Στατιστικά χρήσης

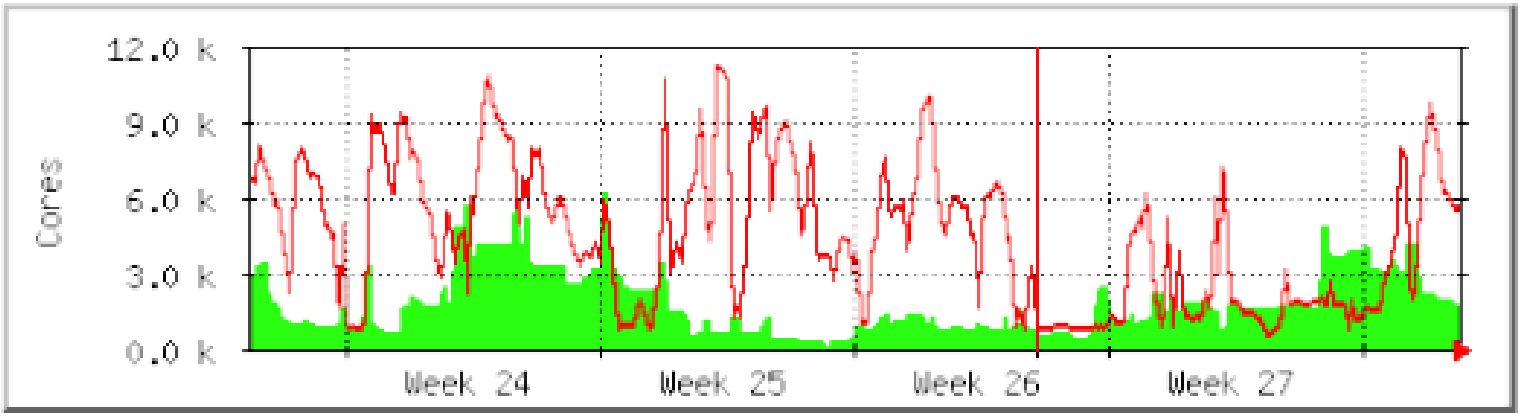
`Yearly' Graph (1 Day Average)



	Max	Average	Current
Cores used by RUNNING jobs	8445 Cores	8118 Cores	8274 Cores
Cores requested by QUEUED jobs	50 kCores	19 kCores	12 kCores

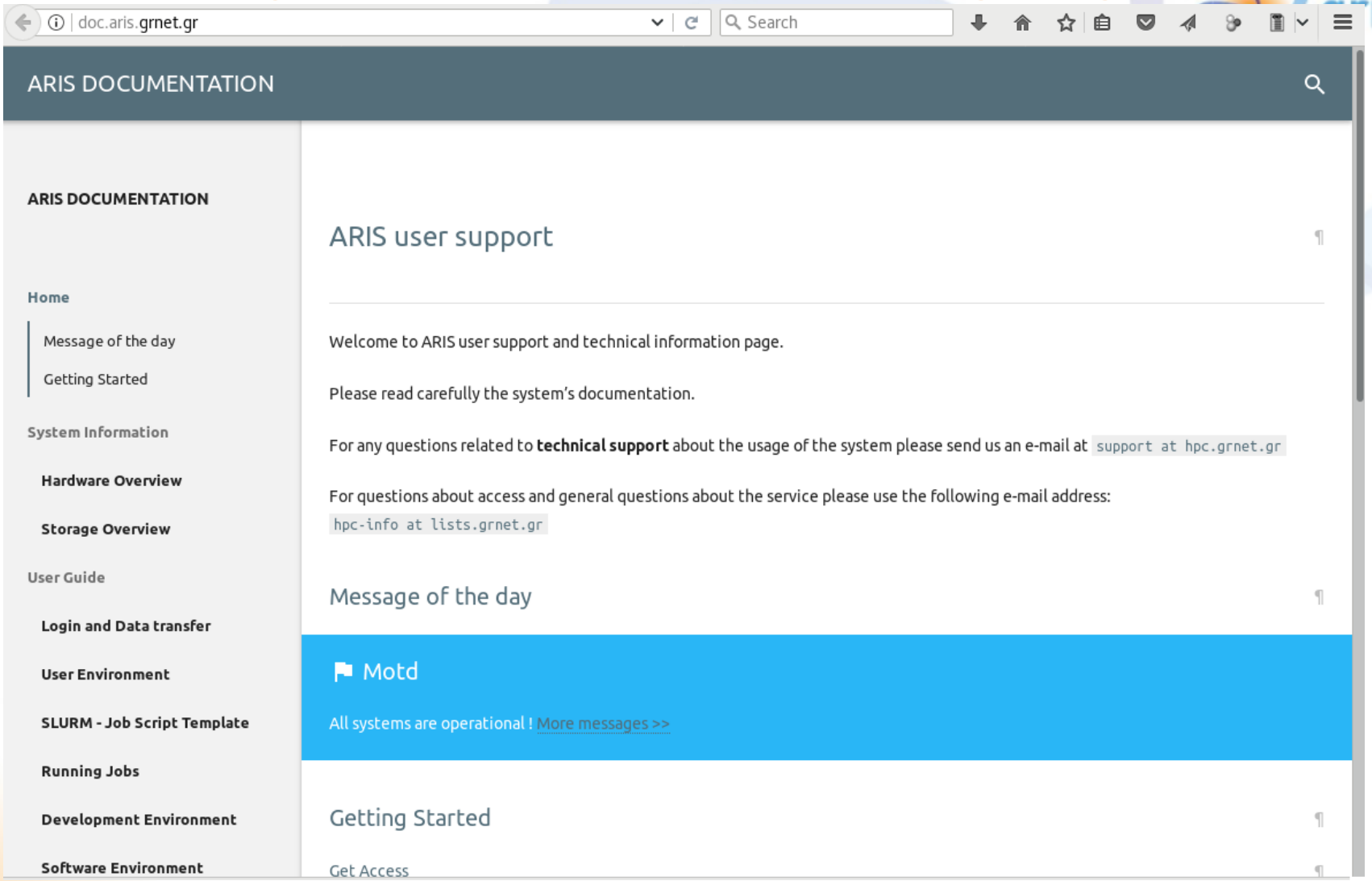
Στατιστικά χρήσης

`Monthly' Graph (2 Hour Average)



	Max	Average	Current
Cores queued due Dependency or Limits	6222 Cores	1797 Cores	1714 Cores
Cores queued due to Priority or Resources	11 kCores	4387 Cores	5257 Cores

Πληροφορίες για το Σύστημα : http://doc.aris.grnet.gr/



Environment Modules.Τι είναι ?

- Για τη χρήση εφαρμογών που δεν προέρχονται από το σύστημα, πρέπει να ρυθμιστούν PATH, LD_LIBRARY_PATH και διάφορες άλλες μεταβλητές περιβάλλοντος για τη λειτουργία των εφαρμογών.
- Συνήθης πρακτική να ρυθμίζονται αυτές οι μεταβλητές είτε γενικά σε κάποιο σύστημα που τρέχει μερικές μόνο εφαρμογές, είτε στο .bashrc του κάθε χρήστη.
- Η κατάσταση περιπλέκεται περισότερο με την ύπαρξη πάνω της μιας versions του ίδιου πακέτου, οι μεταβλητές των οποίων εξαρτώνται από άλλες μεταβλητές.
- Το πακέτο Environment Modules κάνει δυναμική τροποποίηση του περιβάλλοντος χρήστη μέσω των module files.

- Κύριες μεταβλητές περιβάλλοντος που προσαρμόζονται είναι οι PATH, MANPATH, και LD_LIBRARY_PATH, αλλά και μεταβλητές περιβάλλοντος που ενδεχομένως κάθε πακέτο λογισμικού χρειάζεται.
- Κάθε module file περιέχει την πληροφορία που χρειάζεται ώστε να ρυθμίσει τις μεταβλητές περιβάλλοντος για κάποια εφαρμογή.
- Όλα τα modules θέτουν μια μεταβλητή MODULENAMEROOT. Σε modules που αναφέρονται σε βιβλιοθήκες, συνήθως τα include files βρίσκονται στην $MODULENAMEROOT/include και οι βιβλιοθήκες στην $MODULENAMEROOT/lib

- Εάν υπάρχουν εξαρτήσεις ενός πακέτου λογισμικού από άλλα τα οποία επίσης ρυθμίζονται με module file, οι εξαρτήσεις αυτές μπορούν να περιγραφούν και εφόσον το αντίστοιχο module δεν είναι ενεργό είτε το φορτώνει είτε βγάζει μήνυμα λάθους ειδοποιώντας το χρήστη ότι πρέπει πρώτα να φορτώσει τις εξαρτήσεις.
- Σε περιπτώσεις πακέτων τα οποία υπάρχουν σε πάνω από μια έκδοση, υπάρχει ένα module για κάθε έκδοση και ο administrator μπορεί να ορίσει κάποια ως default.

Environment Modules. Χρήση

- Έλεγχος πακέτων που είναι διαθέσιμα μέσω modules

  ```
  module avail
  ```

  ή

  ```
  module -l avail
  ```

- Έλεγχος ενεργών modules

  ```
  module list
  ```

- Απενεργοποίηση όλων των ενεργών modules

  ```
  module purge
  ```

- Απενεργοποίηση συγκεκριμένου module

  ```
  module unload MODULENAME
  ```

- Αλλαγή έκδοσης module

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

- Πληροφορίες για το τι αφορά κάποιο module

```
module whatis MODULENAME/VERSION
```

- Κείμενο Βοήθειας για κάποιο module

```
module help MODULENAME/VERSION
```

Resources Manager - Batch System I

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

Resources Manager - Batch System I

Όταν μια εργασία υποβάλεται σε ένα Batch system :

- Περιγράφονται οι πόροι που χρειάζεται το σύστημα (π.χ. cores, nodes, μνήμη, χρόνος εκτέλεσης
- Το σύστημα κατάγράφει τους πόρους που ζητήθηκαν
- Όταν βρεθούν οι διαθέσιμοι πόροι, ξεκινάει η εκτέλεση της εργασίας.
- Εγγυάται ότι το κάθε run θα έχει πλήρη και **αποκλειστική** πρόσβαση στους πόρους που ζήτησε, π.χ. μνήμη, cores, accelerators κλπ.
- Μπορώ να στείλω π.χ. 1000 runs, τα οποία θα εκτελεστούν χωρίς ταυτόχρονη εκτέλεση στα ίδια resources (μνήμη, cores).

Resources Manager - Batch System II

- Αν κάποιος άλλος χρήστης στείλει run θα πάρει και αυτός το αναλογούν ποσοστό resources χωρίς επικάλυψη.
- Οι πόροι μπορούν να χρησιμοποιηθούν όπως θέλει ο χρήστης
  - Ένα π.χ. MPI run (Η κύρια/προτεινόμενη χρήση)
  - Πολλά σειριακά runs : Αν και μπορεί να χρησιμοποιηθεί με αυτό τον τρόπο, ένα run δεν κερδίζει κάτι από την ύπαρξη π.χ. Infiniband. Ίσως η χρήση της υποδομής Grid : www.hellasgrid.gr ταιριάζει καλύτερα σε τέτοιες εργασίες.

SLURM Scripts

- Ένα SLURM Script περιγράφει τους πόρους που χρειάζεται για να τρέξει η εργασία, όπως επίσης τις εντολές εκτέλεσης της εργασίας.
- http://doc.aris.grnet.gr/scripttemplate/
- Script generator και validator

SLURM Scripts

```
#!/bin/bash
#SBATCH --job-name="testSlurm"   # Όνομα για διαχωρισμό μεταξύ jobs
#SBATCH --error=job.err.%j    # Filename για το stderr
#SBATCH --output=job.out.%j   # Filename για το stdout
                              # Το %j παίρνει την τιμή του JobID
#SBATCH --nodes=200           # Αριθμός nodes
#SBATCH --ntasks=400          # Αριθμός MPI Tasks
#SBATCH --ntasks-per-node=2   # Αριθμός MPI Tasks / node
#SBATCH --cpus-per-task=10    # Αριθμός Threads / MPI Task
#SBATCH --mem=56G             # Μνήμη ανά node  # Από τις 2 επιλογές
#SBATCH --mem-per-cpu=2800M   # Μνήμη ανά core  # προτίνεται η πρώτη.
#SBATCH -A sept2015           # Accounting tag (Θα δοθεί προφορικά αν χρειαστεί)
#SBATCH -t 1-01:00:00         # Ζητούμενος χρόνος DD-HH:MM:SS
#SBATCH -p compute            # partition, compute=default στο ARIS. gpu, phi, fat, taskp
                              # τα εναλλακτικά.
module purge
module load gnu/4.9.2
module load intel/15.0.3
module load intelmpi/5.0.3

if [ x$SLURM_CPUS_PER_TASK == x ]; then            #
  export OMP_NUM_THREADS=1                         #
else                                               # Δεν σβήνουμε αυτά εκτός αν
  export OMP_NUM_THREADS=$SLURM_CPUS_PER_TASK      # ξέρουμε ΑΚΡΙΒΩΣ τι κάνουμε
fi                                                 # και τι συνέπειες μπορεί να εχει.

srun EXECUTABLE ARGUMENTS   # Εδώ το executable και τα πιθανά arguments που παίρνει.
```

Χρήση **srun** για την εκτέλεση των εφαρμογών

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

- Μπορεί να μεταφέρει σε όλα τα tasks τις μεταβλητές περιβάλλοντος που έχουν οριστεί. Με ssh είναι πολύ πιθανό να μη διαδίδονται σε όλα τα tasks οι μεταβλητές περιβάλλοντος.

Επικοινωνία με το SLURM I

- Υποβολή εργασίας

```
sbatch SLURM_JobScript.sh
Submitted batch job 123456
```

- Κατάλογος εργασιών

```
squeue
```

- Κατάλογος εργασιών με περισότερες λεπτομέρειες

```
squeue -o "%.8i %.9P %.10j %.10u %.8T %.5C
%.4D %.6m %.10l %.10M %.10L %.16R"
```

- Ακύρωση εργασίας

```
scancel JobID
```

- Σε κάποιες περιπτώσεις που τα εκτελέσιμα δεν τερματίζονται άμεσα παίρνοντας SIGHUP από το SLURM

Επικοινωνία με το SLURM II

```
scancel -s KILL JobID
```

- Εκτίμηση του πότε θα αρχίσει η εκτέλεση των εργασιών που είναι σε αναμονή για πόρους

```
squeue --start
```

- Πληροφορίες για την τρέχουσα χρήση των πόρων του συστήματος

```
sinfo
```

- Πληροφορίες για την τρέχουσα χρήση των πόρων συγκεκριμένου partition

```
π.χ. sinfo -p gpu
```

SLURM User/Group resource limits I

- Στο SLURM το κάθε account έχει κάποια όρια πόρων που μπορεί να ζητήσει/χρησιμοποιήσει. Τα όρια αυτά εφαρμόζονται σε όλους του χρήστες του account και για όλα τα partitions. Αυτά είναι :
  - Αριθμός Jobs που μπορούν να εκτελούνται ταυτόχρονα, είτε συνολικά είτε ανά partition.
  - Αριθμός Jobs που μπορούν να εκτελούνται ή να βρίσκονται σε αναμονή, είτε συνολικά είτε ανά partition.
  - Μέγιστος αριθμός cores ή nodes που μπορούν να χρησιμοποιηθούν ταυτόχρονα από jobs ενός account, είτε συνολικά είτε ανά partition.
  - Μέγιστη χρονική διάρκεια εκτέλεσης ενός Job, είτε συνολικά είτε ανά partition.
  - Μέγιστος αριθμός nodes ή και cores που μπορεί να ζητήσει ένα Job, είτε συνολικά είτε ανά partition.

SLURM User/Group resource limits II

- Συνολικός αριθμός core hours στη διάρκεια ενός project, είτε συνολικά είτε ανά partition.

Χρήση Accelerators

- GPU
  #SBATCH –partition=gpu
  #SBATCH –gres=gpu:2
  Variable : `SLURM_JOB_GPUS=0,1` και
  `CUDA_VISIBLE_DEVICES=0,1`
- Xeon Phi
  #SBATCH –partition=phi
  #SBATCH –gres=mic:2
  Variable : `OFFLOAD_DEVICES=0,1`

Software

- Live
- Οργάνωση software στο ARIS
- Σχετιζόμενα με ML bundles
- Εγκατάσταση σε user space επιπρόσθετων πακέτων - επίλυση ενδεχόμενων προβλημάτων

Καλές πρακτικές σε DL I

- CPU vs GPU enabled versions
- Native compile vs wheels
  - pip install : Αναλόγως packaging και πολυπλοκότητας, μπορεί είτε να καταβάσει και στήσει precompiled binaries (.whl) ή να κάνει configure/compile/install (.tar.gz) λαμβάνοντας σε ικανοποιητικό βαθμό υπόψιν το σύστημα.
  - Σε υπολογιστικά απαιτητικά python packages, καλό είναι να αποφεύγονται τα wheels.
- Native vs Containers
- Large Data Sets organization / SSD Cache, HDF5, TFRecord, ...
- Διαχείριση datasets με πολλά μικρά files.
- Πιθανό preprocessing σε κάθε read

Καλές πρακτικές σε DL II

- Batch Size, GPU memory
- Use of multiple GPUs - Nodes
- Exclusive access on Resources
- Efficiency / Cost
- Monitor GPU Usage
  - nvidia-smi
  - tensorboard
  - ....
- Προσεκτική χρήση resources. Δεσμεύοντας π.χ. 4 GPU δεν σημαίνει ότι τις χρησιμοποιούμε αν ο κώδικας δεν το κάνει.

Common issues with Python modules I

- Πολλά python modules έχουν εξαρτήσεις που σε μεγάλο βαθμό δεν περιγράφονται σωστά/ακριβώς.
  - Δίνεται στα requirements εξάρτηση από κάποιο άλλο python module χωρίς version ή με κάποια παλιά.
    - Γενικά είναι OK για κάποιο (συνήθως μικρό) διάστημα.
    - Μετά κάποιο διάστημα σε άλλη version Python, ίσως αυτά να περιλαμβάνονται στην core Python library, με ίσως ελαφρώς διαφορετικό implementation. Βάζοντας αυτό το module αρχίζουν τα προβλήματα.
      `typing`, `future` σε αυτή την κατηγορία.
    - Χωρίς version με την τρέχουσα version δουλεύει κανονικά, αύριο που θα έχει βγεί η επόμενη αρχίζουν τα προβλήματα
      `tensorflow 1.15` depends on numpy (αλλά δεν δουλεύει με >= 0.19).

Common issues with Python modules II

- Εξάρτηση από κάποια παλιά version μπορεί να οδηγήσει ακόμα και σε ατέρμονο downgrade διάφορων άλλων python modules.
- Καλή πρακτική : Αν π.χ. στο tensoflow/X.Y.Z χρειαζόμαστε κάποιο πρόσθετο, είναι καλό να χρησιμοποιηθεί μια version που έγινε release το ίδιο χρονικό διάστημα με τη X.Y.Z. Τουλάχιστον σαν σημείο έναρξης, μπορεί και νεώτερες να δουλεύουν - συνήθως μέχρι ενός σημείου. Χρειάζεται έλεγχος ότι δουλεύουν.

- Αρκετά python modules ειδικά σε github δεν συντηρούνται, οπότε μετά από 6 μήνες ίσως έχουν σοβαρά θέματα συμβατότητας.
- Έχει συμβεί σε 2 ημέρες να έχει γίνει ένα ζευγάρι modules ασύμβατο ενώ υπάρχει εξάρτηση...

- Τρόποι εγκατάστασης python modules.

Common issues with Python modules III

- System-wide : Σε μια εγκατάστηση python γενικά χτισμένη γύρω από ένα μεγάλο python module, π.χ. κάποια version tensorflow.
  Τα όποια python modules στήνονται στο σύστημα, ελέγχονται για συμβατότητα, και δεν αλλάζουν.
- User Space 1 : pip3 install –user ModuleName
  Αυτό εγκαθιστά τα (συνήθως binary) files κάτω από την directory $HOME/.local. Αρκετές φορές υπάρχουν προβλήματα, εμφανίζονται άλλες versions κλπ. και πρέπει να ελέγχεται αυτό το PATH αν έχει κάτι εγκατεστημένο.
- User Space 2: Virtual Environments.
- Σειρά αναζήτησης : 1. .local, 2. Virtual Environment, 3. System-wide installation.

Common issues with Python modules IV

- Για σχετικά μικρά python modules που περιλαμβάνουν και κώδικα π.χ. C++, προτιμάτε `pip3 install --no-binary :all:`, κάνει native build από source, δεν είναι τόσο απλό σε αρκετές περιπτώσεις.
- Δεν είναι για μεγάλα python modules όπως π.χ. TF που αφενός έχει και άλλες εξαρτήσεις, αλλά κυρίως χρειάζεται περίπου 6 ώρες σε μια 32-κορη μηχανή με 128 GB μνήμης.

- Binary wheels vs native builds.
  - pip3 install συνήθως κατεβάζει binary files, που έχουν γίνει build σε κάποια μηχανή με συγκεκριμένο OS και system libs. Πολλές φορές αυτά είναι ασύμβατα με το σύστημα στο οποίο προσπαθούμε να τρέξουμε, στην περίπτωση που το σύστημα στο οποίο έγινε build είναι πολύ φρέσκο.

Common issues with Python modules V

- Ειδικά για τις gpus είναι ενδεχόμενο να έχουν γίνει build με πολύ πρόσφατη version CUDA, η οποία χρειάζεται πολύ πρόσφατο NVIDIA driver οπότε δεν δουλεύουν (ή δεν αναγνωρίζουν GPUs και τρέχουν μόνο στη CPU => συνήθως μερικές 100άδες φορές πιο αργά).
- Σε διάφορα python modules όπου δίνονται οδηγίες για venv, anaconda κλπ. περιγράφονται εξαρτήσεις p.x. python 3.9, CUDA 11.1. Αυτές αφορούν τα binary πακέτα. Το build από source έχει εξάρτηση αλλά όχι τόσο strict. Π.χ. torch 1.8.1 λέει για CUDA 10.2 ενώ η πραγματική εξάρτηση είναι 9.x

- Χρήση containers
  - Αν και φαίνεται η λύση σε πολλά, έχει αρκετές περιπλοκές και ίσως αδυναμία χρήσης.
    - Δεν τρέχει κάν γιατί η libc εντός του container image δεν δουλεύει (ή απλά αρνείται) με το kernel του host συστήματος.

Common issues with Python modules VI

- Αν και περιέχει κάποια νέα CUDA version, δεν είναι συμβατή με τον driver της nvidia στο host.
- Ειδικά σε distributed, δεν μπορεί να γνωρίζει κάποιος τι network hardware υπάρχει σε οποιαδήποτε μηχανή, οπότε στο δικτυακό κομμάτι χρησιμοποιείται το πιο συμβατό TCP/IP με σοβαρές επιπτώσεις στο performance ειδικά όταν το bandwidth και κυρίως το latency του δικτύου παίζουν σηματικό ρόλο στο performance.
- Ένα container πρέπει να είναι portable σε πολλές μηχανές => Optimizations περιοριζόμενα σε κάποιο base για hardware.
- Μπορεί να μην δουλεύει κάν αν το base είναι μεγαλύτερο από την τρέχουσα μηχανή
- Αν δουλεύει, μπορεί να είναι μερικές φορές πιο αργό από το native στην τρέχουσα μηχανή

Common issues with Python modules VII

- Πιθανότητα να είναι και portable και efficient στην τρέχουσα μηχανή αρκετά μικρή.