ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΕΦΑΡΜΟΣΜΕΝΩΝ ΜΑΘΗΜΑΤΙΚΩΝ ΚΑΙ ΦΥΣΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

ΤΟΜΕΑΣ ΦΥΣΙΚΗΣ
ΕΡΓΑΣΤΗΡΙΟ ΠΕΙΡΑΜΑΤΙΚΗΣ ΦΥΣΙΚΗΣ ΥΨΗΛΩΝ ΕΝΕΡΓΕΙΩΝ

Ανάπτυξη Εργαλείου για την Εποπτεία των Εργασιών στο Grid

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

Σωτήρη Α. Φραγκίσκου

Επιβλέπων: Θεόδωρος Αλεξόπουλος
Καθηγητής

Αθήνα, Οκτώβριος 2012

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΕΦΑΡΜΟΣΜΕΝΩΝ ΜΑΘΗΜΑΤΙΚΩΝ ΚΑΙ ΦΥΣΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΟΜΕΑΣ ΦΥΣΙΚΗΣ
ΕΡΓΑΣΤΗΡΙΟ ΠΕΙΡΑΜΑΤΙΚΗΣ ΦΥΣΙΚΗΣ ΥΨΗΛΩΝ ΕΝΕΡΓΕΙΩΝ

Ανάπτυξη Εργαλείου για την Εποπτεία των Εργασιών στο Grid

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

Σωτήρη Α. Φραγκίσκου

Επιβλέπων: Θεόδωρος Αλεξόπουλος
Καθηγητής

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την -εισάγετε ημερομηνία-.

.....	.....	.....
Θ. Αλεξόπουλος Καθηγητής	Ε. Γαζής Καθηγητής	Γ. Τσιπολίτης Καθηγητής

Αθήνα, Οκτώβριος 2012.

.....
(Εισάγετε όνομα, αρχικό πατρώνυμου και επίθετο συγγραφέα)
(Εισάγετε τον απονεμηθέντα τίτλο στο συγγραφέα)

.....
(Εισάγετε όνομα, αρχικό πατρώνυμου και επίθετο συγγραφέα)
(Εισάγετε τον απονεμηθέντα τίτλο στο συγγραφέα)

.....
(Εισάγετε όνομα, αρχικό πατρώνυμου και επίθετο συγγραφέα)
(Εισάγετε τον απονεμηθέντα τίτλο στο συγγραφέα)

Θα ήθελα να ευχαριστήσω θερμά το Φώτη Γεωργάτο για τις ατέλειωτες ώρες που μου αφιέρωσε, για τη μετάδοση του ενθουσιασμού του και της αγάπης του για το linux, την python, το Grid και ο,τιδήποτε «κομπιουτερικό» (και φυσικά των πολύτιμων γνώσεών του γύρω από αυτά!).

Θα ήθελα επίσης να ευχαριστήσω ολόψυχα τον καθηγητή μου Θεόδωρο Αλεξόπουλο για την υποστήριξή του όλα αυτά τα χρόνια, χωρίς την οποία δε θα είχα τη δύναμη να ολοκληρώσω την προσπάθειά μου.

Περίληψη

Το qtop είναι ένα εργαλείο που γράφτηκε για να συνοψίσει, με κειμενικό *και* απεικονιστικό τρόπο, την κατάσταση ενός LRMS (Local Resource Management System), μαζί με σχετιζόμενες χρήσιμες πληροφορίες γύρω από συστοιχίες, είτε του Grid ή αυτόνομες.

Η πληροφορία οργανώνεται σε τρία κεφάλαια. Το πρώτο κεφάλαιο κάνει μια συνοπτική περιγραφή του τι είναι το Grid και από τι αποτελείται.

Στο δεύτερο κεφάλαιο περιγράφεται η λειτουργία της εφαρμογής, οι εντολές πάνω στις οποίες βασίζεται, τα αρχεία που παράγει και το βασικότερο, η έξοδός του.

Στο τρίτο κεφάλαιο βρίσκεται η τεκμηρίωση της εφαρμογής qtop, δίνοντας όλα τα απαραίτητα στοιχεία που χρειάζονται για να τη χρησιμοποιήσει ο αναγνώστης, και αν επιθυμεί να την τροποποιήσει, ή και να συνεισφέρει στον κώδικα[1]. Στο τέλος, παρουσιάζονται τα συμπεράσματα από τον προγραμματισμό της εφαρμογής qtop και από την εφαρμογή της σε ένα αριθμό συστοιχιών του Grid.

Τέλος, στο παράρτημα Α βρίσκεται ο πηγαίος κώδικας.

Στην ηλεκτρονική μορφή της εργασίας υπάρχει και παράρτημα Β, στο οποίο παρατίθενται στιγμιότυπα από την εκτέλεση της εφαρμογής πάνω στην υποδομή των εικονικών οργανισμών (VO) ATLAS και SEE.

[1] Ο πηγαίος κώδικας της εφαρμογής βρίσκεται στο δικτυακό τόπο `https://github.com/sfranky/qtop`

Περιεχόμενα

Κεφάλαιο 1

Το Πλέγμα – Grid

1.1 Εισαγωγή

Την τελευταία δεκαετία έχει παρατηρηθεί μια σημαντική αύξηση στη χρήση «κοινών» υπολογιστών (commodity computing) και στις επιδόσεις των δικτύων, κυρίως λόγω του γρηγορότερου υλικού και του πιο εξελιγμένου λογισμικού. Οι κοινές αυτές τεχνολογίες χρησιμοποιήθηκαν για να αναπτυχθούν υπολογιστικά συστήματα υψηλών επιδόσεων αλλά χαμηλού κόστους, γνωστά ως συστοιχίες (clusters), για να επιλύσουν προβλήματα που απαιτούν πολλούς πόρους, σε ένα φάσμα από πεδία εφαρμογών.

Πιο συγκεκριμένα, στην επιστημονική κοινότητα, η διαθεσιμότητα ισχυρών υπολογιστικών πόρων επέτρεψε στους επιστήμονες να διευρύνουν τις εξομοιώσεις και τα πειράματά τους σημαντικά. Τα γρήγορα δίκτυα κατέστησαν δυνατό να γίνεται διαμοιρασμός δεδομένων από όργανα (όπως ο επιταχυντής σωματιδίων LHC στο CERN) και αποτελέσματα πειραμάτων ανάμεσα σε συνεργάτες ανά τον κόσμο σχεδόν ακαριαία, με αποτέλεσμα, την τελευταία δεκαετία, να έχουν αρχίσει ερευνητικοί φορείς να ξεκινούν μεγαλόπνοα προγράμματα που θα διευκολύνουν τη δημιουργία τέτοιων συνεργασιών προς την επίλυση επιστημονικών προβλημάτων μεγάλης κλίμακας[1]

Μια συνέπεια των μεγάλων συνεργασιών και της αυξημένης επεξεργαστικής ισχύος είναι ότι τα δεδομένα που παράγονται και αναλύονται εντός των προγραμμάτων της eScience είναι ογκώδη και ταυτόχρονα εγγενώς διανεμημένα. Επομένως, οι προκλήσεις σε τέτοιου είδους περιβάλλοντα περιστρέφονται γύρω από τα δεδομένα – τη διαχείριση της πρόσβασης σε αυτά, την κατανομή, την επεξεργασία και την αποθήκευση. Οι προκλήσεις αυτές συνεπώς ξύπνησαν την ανάγκη δημιουργίας μιας υπολογιστικής υποδομής, η οποία θα συζεύγει ευρέως κατανεμημένους πόρους όπως είναι οι βάσεις δεδομένων, οι αποθηκευτικοί εξυπηρετητές (storage servers), τα δίκτυα υψηλών ταχυτήτων, οι υπερυπολογιστές και οι συστοιχίες υπολογιστών με σκοπό την επίλυση προβλημάτων μεγάλης κλίμακας. Όλο αυτό οδήγησε σε εκείνο

[1] Συλλογικά, για τα προγράμματα αυτά χρησιμοποιείται ο όρος eScience. Χρησιμοποιείται επίσης ο όρος Big Science, στον οποίο υπονοείται μεγάλο μέγεθος σε ένα ή περισσότερα στοιχεία από τα ακόλουθα: προϋπολογισμός, ανθρώπινο δυναμικό, όργανα, εργαστήρια.

που σήμερα ονομάζουμε Grid computing. Ο όρος Grid είναι δανεισμένος από το electrical power grid, το δίκτυο της ηλεκτρικής ενέργειας, το οποίο παρέχει συνεπή, διάχυτη, αξιόπιστη και διαφανή πρόσβαση στο ηλεκτρικό ρεύμα, ανεξάρτητα από την πηγή. Αν και το Grid είναι ακόμη μακριά από το να βρίσκεται σε πλήρη αναλογία με το ηλεκτρικό δίκτυο, σχεδιάστηκε και αναπτύσσεται με αυτήν την φιλοσοφία υπόψη.

Ένας σχετικός όρος που χρησιμοποιείται ολοένα και περισσότερο σήμερα είναι οι λεγόμενες υπολογιστικές υποδομές κατανεμημένης παραγωγής (production-distributed computing infrastructures)[6], που ορίζονται ως ένα σύνολο υπολογιστικού υλικού και λογισμικού, διανεμημένο σε διαφορετικές τοποθεσίες, και που προορίζεται για χρήση από πολλούς ανθρώπους, οι οποίοι δεν είναι οι ίδιοι οι προγραμματιστές της υποδομής. Υπάρχουν επιστημονικές, ερευνητικές και εμπορικές κατανεμημένες υποδομές. Οι κύριοι εκπρόσωποι κάθε κατηγορίας είναι αντίστοιχα:

- τα TeraGrid, DEISA, Open Science Grid, EGI (πρώην EGEE), NGS,
- τα Grid'5000, PlanetLab, DAS, FutureGrid,
- τα Amazon Web Services και το Microsoft Azure.

1.2 Τι είναι το Grid

Ένα Grid είναι μια υποδομή που περιλαμβάνει την ενσωματωμένη και από κοινού χρήση υπολογιστών, δικτύων, βάσεων δεδομένων και επιστημονικών οργάνων, τα οποία κατέχουν και διαχειρίζονται πολλοί διαφορετικοί οργανισμοί. Οι εφαρμογές του Grid σχετίζονται συχνά με μεγάλες ποσότητες δεδομένων ή/και υπολογιστικούς πόρους που απαιτούν τον ασφαλή διαμοιρασμό τους πέρα από τα όρια των εν λόγω οργανισμών. Αυτό καθιστά τη διαχείριση και ανάπτυξη των εφαρμογών του Grid πολύπλοκο εγχείρημα. Τα λεγόμενα Grid Middleware παρέχουν στους χρήστες απρόσκοπτες επεξεργαστικές δυνατότητες και ομοιόμορφη πρόσβαση στους πόρους του ετερογενούς περιβάλλοντος του Grid. Αρκετά συστήματα και εργαλειοθήκες (toolkits) λογισμικού έχουν αναπτυχθεί, εκ των οποίων τα περισσότερα είναι αποτέλεσμα προγραμμάτων ακαδημαϊκής έρευνας ανά τον κόσμο.

1.3 Τύποι Grid

Υπάρχουν διαφορετικές κατηγορίες Grid που εξυπηρετούν διαφορετικές ανάγκες. Οι τρεις βασικές κατηγορίες πλεγμάτων που υπάρχουν, οι οποίες κλιμακώνονται από απλά συστήματα μέχρι συστήματα υπερυπολογιστών με χιλιάδες επεξεργαστές, είναι οι ακόλουθες:

- Συστοιχία Grid (Cluster Grid): Τα Grid αυτής της κατηγορίας έχουν την απλούστερη μορφή. Αποτελούνται από ένα σύνολο υπολογιστών – hosts που εργάζονται από κοινού. Τα Grid αυτά παρέχουν στους χρήστες ένα μοναδικό

σημείο πρόσβασης στους υπολογιστικούς πόρους ενός προγράμματος ή ενός τμήματος.

- Campus Grid: Τα Grid αυτής της κατηγορίας επιτρέπουν σε πολλαπλά προγράμματα ή τμήματα μέσα σε ένα οργανισμό να μοιράζονται τους υπολογιστικούς του πόρους. Οι οργανισμοί αυτοί μπορούν να χρησιμοποιούν τα Grid αυτά για να εκτελούν εργασίες όπως επαναλαμβανόμενες επιχειρησιακές διαδικασίες, rendering, επεξεργασία δεδομένων και άλλα.

- Global Grid: Τα Grid αυτής της κατηγορίας είναι μια συλλογή campus Grid που ξεπερνούν τα όρια των οργανισμών για να δημιουργήσουν μεγάλα εικονικά συστήματα. Οι χρήστες έχουν πρόσβαση σε υπολογιστική ισχύ που υπερβαίνει κατά πολύ την υπολογιστική ισχύ που είναι διαθέσιμη στους οργανισμούς στους οποίους ανήκουν.

1.4 Αρχιτεκτονική του Grid

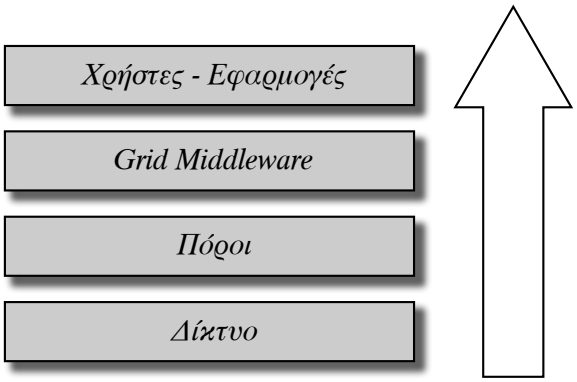
Η αρχιτεκτονική ενός Grid περιγράφεται συχνά σε «επίπεδα», όπου κάθε επίπεδο έχει μια συγκεκριμένη λειτουργία. Τα υψηλότερα επίπεδα επικεντρώνονται σε γενικές γραμμές στο χρήστη, ενώ τα χαμηλότερα επίπεδα είναι περισσότερο υλικο-κεντρικά, εστιάζοντας περισσότερο στους υπολογιστές και τα δίκτυα (σχ. 1.1).

- Το χαμηλότερο επίπεδο είναι το **δίκτυο**, το οποίο συνδέει μεταξύ τους τους διάφορους πόρους του Grid.

- Πάνω από το επίπεδο του δικτύου βρίσκεται το **επίπεδο πόρων**: οι ίδιοι οι πόροι του Grid, όπως οι υπολογιστές, τα αποθηκευτικά συστήματα, οι κατάλογοι ηλεκτρονικών δεδομένων, αισθητήρες και τηλεσκόπια τα οποία είναι συνδεδεμένα με το δίκτυο.

- Το επίπεδο του **Middleware** παρέχει τα εργαλεία που καθιστά δυνατή τη συμμετοχή των διάφορων στοιχείων (εξυπηρετητές, αποθήκευση, δίκτυα, κ.λπ) σε ένα Grid. Το επίπεδο του Middleware θεωρείται ο «εγκέφαλος» πίσω από το Grid.

- Το υψηλότερο δομικό επίπεδο είναι το **επίπεδο εφαρμογών**, το οποίο περιλαμβάνει εφαρμογές για την επιστήμη, τη μηχανική, τις επιχειρήσεις, τα οικονομικά και άλλα, αλλά και πύλες και εργαλειοθήκες ανάπτυξης για την υποστήριξη των εφαρμογών. Αυτό είναι το επίπεδο που οι χρήστες του Grid «βλέπουν» και αυτό με το οποίο αλληλεπιδρούν. Το επίπεδο εφαρμογών περιλαμβάνει συχνά το αποκαλούμενο serviceware, το οποίο εκτελεί λειτουργίες γενικής διαχείρισης, όπως είναι η παρακολούθηση του ποιος παρέχει πόρους του Grid και ποιος τους χρησιμοποιεί.



Σχήμα 1.1

1.5 WLCG

Όπως είδαμε στην ενότητα 1.3, υπάρχουν διάφορα είδη Grid και άρα πολλά διαφορετικά Grid μεταξύ τους. Ένα μεγάλο τέτοιο Grid είναι αυτό που ξεκίνησε στο Ευρωπαϊκό Εργαστήριο Πυρηνικής Φυσικής (CERN) και το οποίο ονομάζεται Worldwide LHC Computing Grid Project, WLCG.

Το πρόγραμμα WLCG είναι μια διεθνής συνεργασία περισσότερων από 170 υπολογιστικών κέντρων σε 36 χώρες, το οποίο συνδέει εθνικές και διεθνείς δομές Grid.

Η αποστολή του προγράμματος WLCG είναι να παρέχει διεθνείς υπολογιστικούς πόρους για την αποθήκευση, διανομή και ανάλυση των περίπου 25 Petabyte (25 εκατομμύρια Gigabyte) δεδομένων που παράγονται κάθε χρόνο από το μεγάλο επιταχυντή αδρονίων (Large Hardon Collider (LHC)) στο CERN στα Γαλλο-ελβετικά σύνορα.

Απαραίτητο για το WLCG είναι το λεγόμενο **Middleware**, που είναι το λογισμικό που οργανώνει και ενοποιεί τους πόρους ενός Grid. Το Middleware αποτελείται από πολλά προγράμματα λογισμικού, περιλαμβάνοντας εκατοντάδες χιλιάδες γραμμές κώδικα. Μαζί, αυτός ο κώδικας αυτοματοποιεί όλες τις αλληλεπιδράσεις «μηχανής–προς–μηχανή» (machine to machine interactions, M2M) που δημιουργούν ένα μοναδικό, απρόσκοπτο Grid.

Εδώ θα γίνει μια σύντομη παρουσίαση της γενικότερης αρχιτεκτονικής του LCG, με έμφαση στα διάφορα μηχανήματα που απαρτίζουν μια συστοιχία Grid, ώστε να καταστεί σαφές πού και πώς εκτελείται η εφαρμογή qtop και ποιών μηχανημάτων την πληροφορία απεικονίζει.

Οι κύριες υπηρεσίες που έχουν υλοποιηθεί στο LCG είναι οι εξής:

- Workload Management System (WMS)
- Data Management System (DMS)
- Information System (IS)
- Authorization and Authentication system

- Monitoring Systems

Για τους σκοπούς της εργασίας αυτής, το κύριο ενδιαφέρον βρίσκεται στην υπηρεσία WMS[2], η οποία είναι υπεύθυνη για τη διαχείρηση των εργασιών που υποβάλλονται από τους χρήστες. Η υπηρεσία αυτή αντιστοιχεί τις απαιτήσεις μιας υποβαλλόμενης εργασίας με τους διαθέσιμους υπολογιστικούς πόρους και προγραμματίζει την εκτέλεση της εργασίας σε μία κατάλληλη συστοιχία υπολογιστών. Στη συνέχεια καταγράφει την πρόοδο εκτέλεσης της εργασίας και επιτρέπει στο χρήστη να ανακτήσει τα δεδομένα εξόδου όταν η εργασία τελειώσει την εκτέλεση της.

Η WMS χρησιμοποιείται στα παρακάτω μηχανήματα σε ένα Cluster Grid:

- User Interface (UI)
- Resource Broker (RB)
- Computing Element (CE)
- Worker Node (WN)
- Proxy Server

Στις επόμενες υποενότητες παρουσιάζονται τα μηχανήματα στα οποία εστιάζεται το ενδιαφέρον μας.

User Interface (UI)

Το **User Interface** είναι το μηχάνημα που επιτρέπει στους χρήστες να έχουν πρόσβαση στις λειτουργίες του Grid. Αποτελεί την «πύλη» για τις υπηρεσίες του Grid, αφού μόνο από εδώ ο χρήστης μπορεί να χρησιμοποιήσει τους πόρους του LCG Grid. Στο μηχάνημα αυτό ο χρήστης πρέπει να έχει προσωπικό λογαριασμό και εγκατεστημένο το προσωπικό του πιστοποιητικό[3].

Computing Element (CE)

Το **Computing Element** αποτελεί το «Grid interface» σε μία συστοιχία υπολογιστών. Με τον όρο αυτό αναφερόμαστε *και* στο μηχάνημα στο οποίο εκτελούνται οι υπηρεσίες του Grid *και* στις ουρές αναμονής που χρησιμοποιούνται από το τοπικό σύστημα. Η ονομασία των ουρών αναμονής των εργασιών έχει την εξής μορφή:

```
1 <hostname>:<port>/<batch_queue_name>
```

[2]για τον αναγνώστη που ενδιαφέρεται να ενημερωθεί και για τις υπόλοιπες υπηρεσίες, βλ. παραπομπή [3], σελ. 19.

[3]Για περισσότερες πληροφορίες, βλ. αναφορά [3]

Το Computing Element διαχειρίζεται μία φάρμα ομογενών υπολογιστικών κόμβων τα οποία ονομάζονται Worker Nodes. Στα Worker Nodes είναι διαθέσιμες όλες οι εντολές και τα Application Programming Interfaces (API) για να εκτελούν πράξεις σε πόρους και δεδομένα του Grid.

Worker Node (WN)

Τα Worker Node είναι οι κόμβοι που εκτελούν τις εργασίες.

Κάθε τόπος (site) που αποτελεί μέρος του LCG διαθέτει ένα ή περισσότερα CE και μια φάρμα από WN που ανήκουν σε αυτά.

1.6 Αποστολή εργασιών στο Grid

Η εμπειρία του Grid και η αλληλεπίδραση με αυτό δεν είναι απλή υπόθεση, δηλαδή, δεν υπάρχει άμεση αλληλεπίδραση όπως υπάρχει με έναν κοινό υπολογιστή (δεν υπάρχει γραφικό περιβάλλον, όπως σε όλους τους σύγχρονους κοινούς υπολογιστές, και ακόμα και στο τερματικό, δεν πληκτρολογούμε απλά μια εντολή και τρέχει κάποιο πρόγραμμα). Για να μπορεί κάποιος να έχει πρόσβαση σ' αυτό, κι έπειτα να είναι σε θέση να τρέξει ένα πρόγραμμα και να πάρει αποτελέσματα, πρέπει πρώτα να περάσει από κάποια διαδικασία, η οποία περιλαμβάνει μια διαδικασία ταυτοποίησης, την εγγραφή σε έναν εικονικό οργανισμό (Virtual Organization, VO), την απόκτηση των σχετικών πιστοποιητικών, το άνοιγμα ενός λογαριασμού σε ένα User Interface.

Η διαδικασία αυτή δεν περιγράφεται εδώ, αφού υποθέτουμε ότι ο αναγνώστης είναι ήδη εγεγραμμένος στο Grid κι έχει ένα λογαριασμό σε κάποιο User Interface[4]. Πάραυτα, εδώ θα περιγραφεί η διαδικασία αποστολής μιας εργασίας στο Grid, αφού είναι κάτι που αποτέλεσε μέρος χρήσης του qtop στην παρούσα εργασία.[5]

1.6.1 Προκαταρκτικά

Συνδεόμαστε στο User Interface στο οποίο έχουμε λογαριασμό.
Δημιουργούμε το παρακάτω shell script, το οποίο θέλουμε να τρέξουμε στο Grid:

```
1 #!/bin/sh
2 pbsnodes -a > pbsnodes_a.txt
3 qstat -q > qstat_q.txt
4 qstat > qstat.txt
```

Λίστα 1.1: *qtop-input.sh*

[4]Για περισσότερες πληροφορίες, μπορεί κανείς να ανατρέξει στη βιβλιογραφία, αναφορά [3].

[5]Θα υποθέσουμε ότι ο αναγνώστης έχει μια εξοικείωση με το λειτουργικό σύστημα Linux και μια στοιχειώδη γνώση των βασικών εντολών του φλοιού Bash.

Προς το παρόν, αγνοούμε το τι κάνει το script. Αυτό που μας ενδιαφέρει είναι ότι καλεί τρεις εντολές, και αποθηκεύει το αποτέλεσμα της κάθεμίας στο αντίστοιχο αρχείο. Το script αυτό θέλουμε να εκτελεστεί όχι στον υπολογιστή στον οποίο βρισκόμαστε (το UI), αλλά σε κάποια Computing Elements.

Έπειτα δημιουργούμε το παρακάτω αρχείο:

```
1 Executable    = "qtop-input.sh";
2 Arguments     = "";
3 StdOutput     = "qtop-input.out";
4 StdError      = "qtop-input.err";
5 InputSandbox = {"qtop-input.sh"};
6 OutputSandbox = {"qtop-input.out","qtop-input.err", "qstat.txt",
      "qstat_q.txt", "pbsnodes_a.txt"};
7 VirtualOrganisation = "atlas";
```

Λίστα 1.2: *qtop-input.jdl*

Το αρχείο αυτό είναι ένα αρχείο γραμμένο στη γλώσσα jdl (Job Description Language), η οποία χρησιμοποιείται για να περιγράψουμε την εφαρμογή που θέλουμε να τρέξουμε στο Middleware.
Στη γραμμή 1 αναφέρεται το script το οποίο θέλουμε να τρέξουμε στο Grid (το αμέσως προηγούμενο shell script).
Στη γραμμή 2 δίνουμε τυχόν παραμέτρους (εδώ καμμία).
Στις γραμμές 3 και 4 ορίζουμε τα αρχεία στα οποία θα γραφεί, αντίστοιχα, η έξοδος και τα τυχόν σφάλματος από την εκτέλεση της εφαρμογής μας.
Στις γραμμές 5 και 6 ορίζουμε αντίστοιχα τα αρχεία που χρειαζόμαστε να ανεβάσουμε από το UI στους υπολογιστές του Grid και τα αρχεία που θέλουμε να επιστραφούν μετά την εκτέλεση από το Grid στο UI.
Στη γραμμή 7 ορίζουμε τον εικονικό οργανισμό στον οποίο θέλουμε να τρέξει η εφαρμογή μας.

1.6.2 Αποστολή εργασίας

Τρέχουμε την εντολή:

```
1 glite-wms-job-list-match -a /qtop-input.jdl > CEs
```

η οποία βρίσκει τα Computing Element που ταιριάζουν με τα κριτήρια της εφαρμογής μας και ανήκουν στον ίδιο VO με μας, κι έπειτα σώζει την πληροφορία στο αρχείο ∼/CEs

Τώρα, αν θέλαμε να στείλουμε την εφαρμογή μας σε ένα μόνο από τα παραπάνω Computing Element, απλά θα τρέχαμε την εντολή:

```
1 glite-wms-job-submit -o jobID -a qtop-input.jdl
```

και στο αρχείο jobID θα αποθηκευόταν ένα αναγνωριστικό για την εργασία που στείλαμε, ώστε να μπορούμε να την παρακολουθήσουμε, και αργότερα να την ανακτήσουμε.

Λόγω της φύσης όμως της εφαρμογής που αναπτύξαμε, το αρχείο αυτό θέλουμε να εκτελεστεί όχι σε ένα, αλλά σε *όλα* τα διαθέσιμα Computing Element του VO μας. Για το σκοπό αυτό φτιάχνουμε το παρακάτω script:

```
1  #!/bin/sh
2  JOBNAME='qtop-input'
3  JOBDIR=$HOME/jobIDs/$JOBNAME-job
4  while read line; do
5  fname3=$line
6  fname2="${fname3//\//_}"
7  fname="${fname2//:/-}"
8  #glite-wms-job-submit -r <CeId=Queue Name> -o <filepath to save
       the jobId to> -a jobfilename.jdl
9  glite-wms-job-submit -r $line -o $JOBDIR/$fname -a $JOBNAME.jdl
10 done <$HOME/CEs
```

Λίστα 1.3: *jobsubmits.sh*

Το script αυτό έχει την εξής λειτουργία: Διαβάζει κάθε γραμμή του αρχείου CE που αποθηκεύσαμε λίγο νωρίτερα (δηλαδή διαβάζει τη διεύθυνση κάθε Computing Element που κρίθηκε κατάλληλο για την εκτέλεση της εργασίας μας) και έπειτα στέλνει την εργασία σε καθένα από αυτά τα Computing Element. Έπειτα αποθηκεύει το αναγνωριστικό για κάθε εργασία σε κατάλληλο φάκελο, με όνομα αρχείου το όνομα της ουράς του CE[6].

1.6.3 Εποπτεία και παραλαβή εργασίας

Με την εντολή:

```
1     glite-wms-job-status -i jobID
```

πληροφορούμαστε για την κατάσταση της εργασίας που έχουμε στείλει. Όταν γράφει Done (Success), θα μπορούμε να πάρουμε τα αποτελέσματα της εργασίας. Στην προκειμένη περίπτωση βέβαια, μας ενδιαφέρει να πάρουμε τα αποτελέσματα *όλων* των εργασιών που στείλαμε, άρα πρέπει να τροποποιήσουμε κατάλληλα την εντολή: script.

```
1     glite-wms-job-status /jobID_directory/* |grep Current
```

[6]Στις γραμμές 6 και 7 μετατρέπουμε την άνω κάτω τελεία και την ανάποδη κάθετο σε παύλα και κάτω παύλα αντίστοιχα, διότι οι χαρακτήρες αυτοί δεν επιτρέπεται να χρησιμοποιηθούν ως μέρος της ονομασίας ενός αρχείου ή ενός φακέλου

Το `|grep Current` στο τέλος της εντολής απλά φιλτράρει την έξοδο και κρατάει μόνο τη γραμμή από κάθε αναφορά που παρουσιάζει την κατάσταση της εργασίας.

Όταν μια εργασία έχει εκτελεστεί, μπορούμε να πάρουμε τα αποτελέσματα με την παρακάτω εντολή:

```
glite-wms-job-output -i jobID --dir ./
```

Η εντολή αυτή και πάλι αφορά *ένα* αποτέλεσμα, οπότε εμείς για να πάρουμε τα αποτελέσματα από την εκτέλεση της εργασίας σε όλα τα Computing Element, δημιουργούμε και εκτελούμε το παρακάτω script:

```
#!/bin/sh
JOBNAME='qtop-input'
JOBDIR=$HOME/jobIDs/$JOBNAME-job

while read line;
do
fname3=$line
fname2="${fname3//\//_}"
fname="${fname2//:/-}"
glite-wms-job-output --dir $HOME/outputs/atlas/$fname-output/ `
   sed -n '02p' $JOBDIR/$fname` ;
done <$HOME/CEs
```

Λίστα 1.4: *joboutputs.sh*

το οποίο, κατ' αντιστοιχία, διαβάζει τη διεύθυνση του καθενός Computing Element, ζητάει τα αποτελέσματα της εκτέλεσης της εργασίας από εκεί, και σώζει τα αρχεία που επιστρέφονται σε κατάλληλο φάκελο.

Κεφάλαιο 2

Η εφαρμογή

2.1 Εισαγωγικά

2.1.1 Υπόβαθρο

Η εφαρμογή που παρουσιάζεται σ' αυτή την εργασία βασίζεται στην ομώνυμη εφαρμογή που είχε γράψει λίγο παλιότερα ο Φώτης Γεωργάτος σε μορφή shell script[7]. Ο κύριος λόγος που οδήγησε στον επαναπρογραμματισμό του qtop από την αρχή στη γλώσσα python είναι ότι υπήρχαν ανάγκες να συμπληρωθεί σε χαρακτηριστικά, κάτι που καθίστατο ολοένα και πιο δύσκολο με τον κώδικα shell. Η καινούργια έκδοση της εφαρμογής είναι αρκετά πιο εύκολο να επεκταθεί (για να συνεργάζεται, για παράδειγμα, με περισσότερα Local Resource Management Systems (LRMSs) πέρα από το torque/PBS, όπως είναι η Oracle/Sun Grid Engine (SGE/GE), τα LSF, LoadLeveler, OAR, SLURM κ.α. Επίσης τα περισσότερα από τα bugs που ταλαιπωρούσαν την παλιά έκδοση έχουν εξαλειφθεί.

2.1.2 Στόχοι του qtop

Το qtop είναι ένα εργαλείο που γράφτηκε για να συνοψίσει, με κειμενικό *και* απεικονιστικό τρόπο, την κατάσταση ενός LRMS , μαζί με κάποιες σχετιζόμενες πληροφορίες γύρω από συστοιχίες, είτε του Grid ή αυτόνομες. Συγκεντρώνει τις πληροφορίες που αναφέρονται στην υποενότητα 2.4 και τις οργανώνει συμπυκνωμένα αλλά παραστατικά, για γρήγορη πρόσβαση.

Ένα στοιχείο που θεωρήθηκε σημαντικό και λήφθηκε υπόψη κατά το σχεδιασμό του προγράμματος είναι η επεκτασιμότητα. Το λογισμικό και το υλικό του Grid εξελίσσονται με ταχείς ρυθμούς και το qtop πρέπει να είναι σε θέση να μπορεί να συμπεριλάβει αργότερα νέες τεχνολογίες συστημάτων LRMS, όπως αυτά που αναφέρθηκαν στην προηγούμενη παράγραφο.

Επίσης, είναι επιθυμητό η πληροφορία που συλλέγει το qtop να μπορεί να επαναχρησιμοποιηθεί από άλλους προγραμματιστές ή διαχειριστές συστημάτων, ή

να αξιοποιηθεί και να απεικονιστεί διαφορετικά. Για το σκοπό αυτό, το qtop δεν επεξεργάζεται απευθείας την έξοδο των αρχείων εισόδου `pbsnodes -a`, `qstat`, και `qstat -q` για την ανάκτηση της χρήσιμης πληροφορίας, αλλά τη μετατρέπει πρώτα σε μια πιο ευανάγνωστη και φιλική προς τον αναγνώστη μορφή, που ονομάζεται YAML, και που είναι πλήρως συμβατή με τη μορφή JSON[1]. Η πληροφορία αυτή αποθηκεύεται εκ νέου σε ενδιάμεσα αρχεία (`pbsnodes_a.yaml`, `qstat_q.yaml` και `qstat.yaml` αντίστοιχα), για μεταγενέστερη εκμετάλλευση. Από τα νέα αυτά αρχεία πλέον, αρχίζει να κάνει τους απαραίτητους υπολογισμούς για να απεικονίσει την πληροφορία κατάλληλα.

2.2 Αρχεία εισόδου qtop

Για να παρέχει την απεικόνισή του, το qtop αξιοποιεί την πληροφορία που παρέχουν τρεις εντολές:

- η `pbsnodes -a`,
- η `qstat`,
- η `qstat -q`.

Στις επόμενες υποενότητες αναλύεται η έξοδος που παρέχει καθεμία από τις παραπάνω εντολές.

2.2.1 pbsnodes -a

Η εντολή `pbsnodes`, ενώ είναι χρήσιμη για τη διαχείριση του συστήματος PBS, μπορεί επίσης να παρέχει χρήσιμες πληροφορίες στο χρήστη του PBS. Η `pbsnodes -a` παρέχει σε λίστα όλους τους κόμβους PBS, τα χαρακτηριστικά τους, καθώς και την κατάσταση των εργασιών. Η εντολή αυτή θα χρησιμοποιηθεί για να ανακτηθεί μια λίστα με έγκυρα χαρακτηριστικά των μηχανημάτων της κάθε συστοιχίας. Στη λίστα 2.1 φαίνεται ένα απόσπασμα από την έξοδο της εντολής `pbsnodes -a`.

Από εδώ το qtop συλλέγει τις εξής πληροφορίες:

- την ονομασία του Worker Node (το Fully Qualified Domain Name, FQDN) και την αρίθμησή του,
- την κατάστασή του,
- τον αριθμό πυρήνων (np, number of processors),
- ποιες εργασίες έχουν ανατεθεί σε ποιον κόμβο.

[1] βλ. και υποενότητα 2.3

Με τον όρο «κατάσταση», εννοείται το αν ο κόμβος είναι πλήρης από εργασίες (job-exclusive), ή αν είναι μη λειτουργικός (down) ή –εκούσια– εκτός λειτουργίας (offline), οπότε και δεν δέχεται περαιτέρω εργασίες. Αν η κατάστασή του δεν περιγράφεται από κανένα από τα παραπάνω, οπότε ο κόμβος είναι λειτουργικός και ελεύθερος ή απλά (όχι αποκλειστικά) επιβαρυμένος με κάποια εργασία, τότε αναφέρεται ως ελεύθερος (free).

Για παράδειγμα, στη λίστα 2.1, το qtop συλλέγει τις εξής πληροφορίες:

- η τοπική ονομασία του μηχανήματος (localhost) είναι `wn001`, το parent domain name του είναι `kallisto.hellasgrid.gr`. Το πρόθεμα «wn» αποθηκεύεται ξεχωριστά.
- η κατάστασή του είναι job–exclusive, δηλαδή ο κόμβος είναι πλήρης από εργασίες,
- διαθέτει δύο πυρήνες,
- και τέλος του έχουν ανατεθεί δύο εργασίες, η πρώτη με `jobID` 785592 στον πυρήνα 0 και η δεύτερη με `jobID` 785595 στον πυρήνα 1.

```
1  wn001.kallisto.hellasgrid.gr
2   state = job-exclusive
3   np = 2
4   properties = lcgpro
5   ntype = cluster
6   jobs = 0/785592.cream01.kallisto.hellasgrid.gr, 1/785595.
       cream01.kallisto.hellasgrid.gr
7   status = rectime=1349967443,varattr=,jobs=778373.cream01.
       kallisto.hellasgrid.gr 785592.cream01.kallisto.hellasgrid
       .gr 785595.cream01.kallisto.hellasgrid.gr,state=free,
       netload=19508811914207,gres=,loadave=2.15,ncpus=4,physmem
       =2058928kb,availmem=2833520kb,totmem=4067044kb,idletime
       =4647201,nusers=3,nsessions=4,sessions=780 920 2947 3081,
       uname=Linux wn001.kallisto.hellasgrid.gr 2.6.18-194.11.4.
       el5 #1 SMP Tue Sep 21 06:46:41 EDT 2010 x86_64,opsys=
       linux
8   gpus = 0
```

Λίστα 2.1: *Απόσπασμα από την έξοδο της εντολής* `pbsnodes -a`

2.2.2 qstat

Η εντολή qstat παρέχει πληροφορίες για την κατάσταση των εργασιών, των ουρών, ή ενός batch server. Η ζητούμενη κατάσταση γράφεται στην έξοδο.

```
1 node1:/test> qstat
2 Job id          Name             User     Time Use S Queue
```

```
3  -------------- ---------------  -------- -------- - -----
4  784110.cream01 cream_110958071  see187   374:00:3 R see
5  784485.cream01 cream_479689766  see036   138:53:3 R see
6  784494.cream01 cream_782698536  see187   204:03:4 R see
7  784580.cream01 cream_633344337  see187   254:48:5 R see
8  784883.cream01 cream_651406758  compc023 24:28:47 R compchem
9  784950.cream01 cream_372381376  see192   00:00:00 R see
10 785111.cream01 cream_974362714  see187   00:00:00 R see
11 785546.cream01 cream_342185948  compc129 05:20:15 R compchem
12 785563.cream01 cream_775538827  compc129 05:22:09 R compchem
13 785585.cream01 cream_701131251  compc129 05:21:06 R compchem
14 785587.cream01 cream_779716777  compc129 05:20:59 R compchem
15 785589.cream01 cream_490530377  compc165 00:39:14 R compchem
16 785592.cream01 cream_462296854  compc129 05:18:26 R compchem
17 785593.cream01 cream_317513201  compc129 05:18:52 R compchem
18 785594.cream01 cream_747551807  compc129 05:20:02 R compchem
19 785595.cream01 cream_592827680  compc129 05:18:55 R compchem
20 785596.cream01 cream_674157368  compc129 05:17:56 R compchem
21 785603.cream01 cream_210186715  see192   00:00:00 R see
22 785604.cream01 cream_389980515  compc129 05:19:00 R compchem
23 785606.cream01 cream_783085600  compc129 05:16:52 R compchem
24 785611.cream01 cream_196829148  compc129 05:16:59 R compchem
25 785612.cream01 cream_532087070  compc129 05:15:30 R compchem
26 785613.cream01 cream_296044122  compc165 00:31:14 R compchem
27 785614.cream01 STDIN            compc165 01:09:27 R compchem
28 785621.cream01 STDIN            compc129 01:20:01 R compchem
29 785622.cream01 cream_184701588  compc129 00:45:22 R compchem
30 785623.cream01 cream_749791378  compc129 00:45:31 R compchem
31 785625.cream01 cream_136113140  compc129 00:22:16 R compchem
32 785639.cream01 cream_257257844  compc129 00:00:16 R compchem
33 785654.cream01 STDIN            biome138 00:00:00 R biomed
34 785661.cream01 cream_174468395  see162         0 Q see
35 785662.cream01 cream_413314821  see162         0 Q see
36 785665.cream01 cream_143012951  see162         0 Q see
37 785666.cream01 cream_511136412  see162         0 Q see
38 785669.cream01 STDIN            see162         0 Q see
39 785670.cream01 cream_694313146  see162         0 Q see
40 785689.cream01 cream_649866162  see119         0 R see
41 785690.cream01 STDIN            see119         0 Q see
42 785691.cream01 STDIN            biome155 00:30:02 R biomed
```

Λίστα 2.2: *Απόσπασμα από την έξοδο της εντολής* `qstat`**escapechar**

Οι πληροφορίες που το qtop αποθηκεύει είναι:

- το αναγνωριστικό της εργασίας που έχει δοθεί από το PBS,
- το όνομα της εργασίας, όπως έχει δοθεί από το χρήστη που την υπέβαλε,
- ο κάτοχος της εργασίας,

- η κατάσταση της εργασίας
  - **E :** (Exiting) - έξοδος από την εργασία αφού έχει τρέξει,
  - **H**: (Held) - η εργασία έχει κατακρατηθεί,
  - **Q**: (Queued) - η εργασία είναι σε αναμονή, είναι επιλέξιμη να τρέξει, ή είναι δρομολογημένη,
  - **R**: (Running) - η εργασία τρέχει.

2.2.3 qstat -q

Η ίδια εντολή `qstat`, όταν δοθεί με την παράμετρο `-q`, ορίζει ότι η αίτηση γίνεται για την κατάσταση ουράς, και ζητάει να εμφανιστεί σε μια εναλλακτική μορφή παρουσίασης. Το αποτέλεσμα φαίνεται παρακάτω:

```
1
2  server: cream01.kallisto.hellasgrid.gr
3
4  Queue    Memory CPU Time Walltime Node  Run Que Lm  State
5  -------- ------ -------- -------- ----  --- --- --  -----
6  gridcc     --   48:00:00 72:00:00   --    0   0 11   E R
7  sixt       --      --    72:00:00   --    0   0 11   D R
8  biomed     --   48:00:00 72:00:00   --    2   0 11   E R
9  planck     --   48:00:00 72:00:00   --    0   0 11   E R
10 alice      --   48:00:00 72:00:00   --    0   0 11   E R
11 ops        --   48:00:00 72:00:00   --    0   0 12   E R
12 cms        --   48:00:00 72:00:00   --    0   0 11   E R
13 compchem   --   48:00:00 72:00:00   --   22   0 11   E R
14 hgdemo     --   48:00:00 72:00:00   --    0   0 11   E R
15 see        --   48:00:00 72:00:00   --    8   7 11   E R
16 scier      --   48:00:00 72:00:00   --    0   0 11   E R
17 dorii      --   48:00:00 72:00:00   --    0   0 11   E R
18 esr        --   48:00:00 72:00:00   --    0   0 11   E R
19 seeops     --   48:00:00 72:00:00   --    0   0 11   E R
20 nwchem     --   48:00:00 72:00:00   --    0   0 11   E R
21 magic      --   48:00:00 72:00:00   --    0   0 11   E R
22 dteam      --   48:00:00 72:00:00   --    0   0 12   E R
23 envir      --   48:00:00 72:00:00   --    0   0 11   E R
24 atlas      --   48:00:00 72:00:00   --    0   0 11   E R
25 meteo      --   48:00:00 72:00:00   --    0   0 11   E R
26 complex    --   48:00:00 72:00:00   --    0   0 11   E R
27 seismo     --   48:00:00 72:00:00   --    0   0 11   E R
28                                        ----- ---
29                                           32   7
```

Λίστα 2.3: *Απόσπασμα από την έξοδο της εντολής* `qstat -q`

Οι πληροφορίες που το qtop αποθηκεύει είναι:

- το όνομα της ουράς (Queue),
- ο αριθμός εργασιών στην ουρά που βρίσκονται στην κατάσταση «τρέχουσα» (“running”), ο αριθμός των εργασιών στην ουρά στην κατάσταση «σε αναμονή» (“queued”),
- ο μέγιστος αριθμός (όριο, Limit) εργασιών που μπορούν να εκτελεστούν στην ουρά ταυτόχρονα,
- η κατάσταση της ουράς, η οποία δίνεται από ένα ζεύγος γραμμάτων:
  - είτε το γράμμα **E** εάν η ουρά είναι ενεργοποιημένη (Enabled) ή το **D** αν είναι απενεργοποιημένη (Disabled), και
  - είτε το γράμμα **R** αν η ουρά είναι τρέχουσα (Running) ή το **S** αν είναι σταματημένη (stopped),
- το σύνολο των τρέχουσων εργασιών,
- το σύνολο των εργασιών σε αναμονή.

2.3 Αρχεία YAML

Όπως είδαμε στην υποενότητα 2.1.2, το qtop συλλέγει την πληροφορία που παίρνει από την έξοδο των εντολών `pbsnodes -a`, `qstat`, και `qstat -q` και την αποθηκεύει σε τρία αρχεία, σε μια μορφή που ονομάζεται YAML. Η YAML είναι μια μορφή σειριοποίησης δεδομένων αναγνώσιμη από τον άνθρωπο που βασίζεται σε ιδέες από γλώσσες προγραμματισμού όπως είναι η C, Perl, και η Python, και σε ιδέες από την XML[10], ενώ είναι επίσης συμβατή με το στάνταρντ JSON[2]. Στις λίστες 2.4, 2.5 και 2.6 φαίνεται ένα απόσπασμα-παράδειγμα από κάθε αρχείο YAML που δημιουργεί και αποθηκεύει το qtop.

```
1  domainname: wn002dpp.nipne.ro
2  state: b
3  np: 4
4  - core: 0
5    job: 1159732
6  - core: 1
7    job: 1159733
8  - core: 2
9    job: 1159921
10 gpus: 0
11
12 domainname: wn053dpp.nipne.ro
13 state: d
14 np: 8
```

[2] Για περισσότερες πληροφορίες βλ. `http://en.wikipedia.org/wiki/JSON`

```
gpus: 0

...
```

Λίστα 2.4: *Απόσπασμα από το αρχείο pbsnodes_a.yaml*

```
---
JobId: 1042112
UnixAccount: pilatl01
S: Q
Queue: atlas
...
---
JobId: 1085318
UnixAccount: pilatl01
S: Q
Queue: atlas
...
```

Λίστα 2.5: *Απόσπασμα από το αρχείο qstat.yaml*

```
- QueueName: ifops
  Running: 0
  Queued: 0
  Lm: --
  State: E R

- QueueName: hone
  Running: 1
  Queued: 0
  Lm: --
  State: E R

- QueueName: atlas
  Running: 199
  Queued: 195
  Lm: --
  State: E R

- QueueName: ops
  Running: 0
  Queued: 0
  Lm: --
  State: E R

- QueueName: dteam
  Running: 0
  Queued: 0
  Lm: --
```

```
29   State: E R
30
31 ---
32 Total Running: 200
33 Total Queued: 195
```

Λίστα 2.6: *Αρχείο qstat_q.yaml*

2.4 Οι τομείς του qtop

Οι πληροφορίες που αναφέρει το qtop όταν εκτελείται χωρίζονται σε τρεις τομείς:

Λογιστική σύνοψη (Accounting summary)

- αριθμός κόμβων (διαθέσιμοι/σύνολο)
- αριθμός πυρήνων (διαθέσιμοι/σύνολο)
- αριθμός εργασιών (τρέχουσες/σε αναμονή)
- ονόματα των ουρών, με τις τρέχουσες/σε αναμονή εργασίες για την καθεμία.

Πληρότητα κόμβων (Worker Nodes occupancy)

- αναγνωριστικό κόμβου, Node ID (εικονικό ή αληθινό όνομα κόμβου).

  Η εν λόγω πληροφορία διαβάζεται κατακόρυφα και δεν αναφέρεται απαραίτητα στην αρχική αρίθμηση του κόμβου, αλλά σε μία αρίθμηση που μπορεί να προέρχεται από αναδιάταξη (remapping)

- Κατάσταση του κόμβου (Node state)

  Συμβολίζεται με το πρώτο γράμμα μίας εκ των παρακάτων καταστάσεων: job-exclusive, busy, offline, down. Το σύμβολο '–' δηλώνει ότι ο κόμβος είναι ελεύθερος.

- πίνακας κατανομής εργασιών

  Σε μία διάταξη–πίνακα απεικονίζονται κατακόρυφα οι πυρήνες και οριζόντια οι κόμβοι (Worker Nodes), που στην ουσία είναι και ο τρόπος με τον οποίο το LRMS κάνει το σχεδιασμό των πόρων.

Πληροφορίες λογαριασμού χρηστών (User account information)

- «id», που είναι ένα σύμβολο που αναπαριστά ένα μοναδικό λογαριασμό unix
- οι εργασίες του κάθε χρήστη, με αναφορά ξεχωριστά σε αυτές που είναι τρέχουσες σε σχέση με τις συνολικές (συμπεριλαμβάνοντας τις περατωμένες εργασίες)

- το όνομα του λογαριασμού unix
- Grid certificate DN

  Το παραπάνω είναι χρήσιμο σε συστοιχίες συνδεδεμένες στο Grid. Δείχνει σε ποιόν ανήκει ο λεγόμενος λογαριασμός pool στην παρούσα στιγμή[3].

  Σημείωση: το χαρακτηριστικό αυτό μπορεί να τρέξει μόνο έχοντας προνόμια χρήστη root, εξαιτίας της εξάρτησης από την εντολή `edg-mkgridpool --list`.

Οι τρεις παραπάνω τομείς φαίνονται[4][5] αναλυτικότερα αντίστοιχα στις λίστες 2.7, 2.8 και 2.9.

```
===> Job accounting summary <===
01:40:26.482039 WORKDIR = to be added
Usage Totals:  42/48   Nodes | 200/316  Cores |
200+195 jobs (R + Q) reported by qstat -q
Queues: |  ifops: 0+0 | hone: 1+0 | atlas: 199+195 |
ops: 0+0 | dteam: 0+0 | * implies blocked
```

Λίστα 2.7: *πρώτος τομέας*

```
===> Worker Nodes occupancy <===
0000000001111111111222222222233333333344444444445={_Worker_}
1234567890123456789012345678901234567890123456789O={__Node__}
??bbbjbbjb-jjbjbd----do--j-jj-jbbjbbbjobbdbbbjbbbd=Node state
??1_001000_10000__50____0010000__00__0______110_1_=Core0
??10_1__0__1010____00__000_1001__0_1_0__1_0_11__1_=Core1
??11010000_00_01___01__0_0101001100_10_01_01_10_1_=Core2
??_1000101_00100___81__00010070_1001_0_____0_1_11_=Core3
??##################00__000010_11_00110____0_11011_=Core4
??##################_0__00000100__001_0__1_01110_1_=Core5
??##################00__00000000_10___0_01___1111__=Core6
??##################00__000000001100_10_0__00110___=Core7
```

Λίστα 2.8: *δεύτερος τομέας*

```
===> User accounts and pool mappings <===
 id |  R   +   Q  /  all |    unix account |
  0 |  127 +   56 /  183 |        prdatl28 |
  1 |   70 +   90 /  160 |        prdatl73 |
  2 |    0 +   22 /   22 |        pilatl15 |
  3 |    0 +   20 /   20 |        pilatl11 |
```

[3]Το εν λόγω χαρακτηριστικό δεν είχε πραγματοποιηθεί την ημέρα περάτωσης της εργασίας, όσον αφορά την τελευταία έκδοση του qtop.

[4]Το παράδειγμα είναι ελαφρώς τροποποιημένο ώστε να χωρέσει ολόκληρο στη σελίδα.

[5]Στην ηλεκτρονική μορφή της εργασίας περιλαμβάνεται ένα πιο πλήρες έγχρωμο παράδειγμα στο Παράρτημα Β, βλ. σχήμα Β΄.1

```
7   4 |    0 +    6 /    6 |        pilatl01 |
8   5 |    1 +    0 /    1 |         hone031 |
9   6 |    0 +    1 /    1 |        sgmatl54 |
10  7 |    1 +    0 /    1 |        atlas102 |
11  8 |    1 +    0 /    1 |        atlas114 |
```

Λίστα 2.9: *τρίτος τομέας*

Σημείωση: Το qtop εδώ έχει τρέξει σε μια συστοιχία του Horia Hulubei National Institute of Physics and Nuclear Engineering - IFIN HH στη Ρουμανία, στις 13/10/12.

2.4.1 Ο πίνακας του qtop

Η «καρδιά» της εφαρμογής βρίσκεται στον δεύτερο τομέα (βλ. λίστα 2.8), που είναι ουσιαστικά ένας πίνακας διαστάσεων $(1+n)\times x$, όπου n ο μέγιστος αριθμός πυρήνων, x ο αριθμός των κόμβων της συστοιχίας, και το 1 αναφέρεται στη γραμμή με την κατάσταση κάθε κόμβου. Στο συγκεκριμένο παράδειγμα, βλέπουμε με μια γρήγορη ματιά ότι υπάρχουν 50 κόμβοι (Worker Node) (διαβάζοντας την τελευταία στήλη, γραμμές 2 και 3, κατακόρυφα). Στην πραγματικότητα, αυτό δεν ισχύει γενικά (και ούτε στο συγκεκριμένο παράδειγμα)· ο λόγος θα φανερωθεί στη συνέχεια. Η επόμενη γραμμή (γραμμή 4) απεικονίζει συνοπτικά την κατάσταση του κάθε κόμβου.

```
1 ??bbbjbbjb-jjbjbd----do--j-jj-jbbjbbbjobbdbbbjbbbd=Node state
```

Τα πρώτα ερωτηματικά υποδηλώνουν ότι οι κόμβοι 1 και 2 στην πραγματικότητα δεν υπάρχουν· δηλαδή η αρίθμηση ξεκινάει από τον αριθμό 3. Όπου υπάρχει ερωτηματικό, συμπεραίνουμε ότι ο αντίστοιχος κόμβος δεν υπάρχει (είτε δεν υπήρξε ποτέ είτε έχει αφαιρεθεί εξ' ολοκλήρου από τη συστοιχία). Επομένως, οι συνολικοί κόμβοι είναι $50-2=48$. Επειδή πολλές φορές οι ανύπαρκτοι κόμβοι είναι πολυάριθμοι, το qtop παρέχει την επιλογή αυτοί να μην εμφανίζονται καθόλου, και να γίνεται μια αναδιάταξη στην αρίθμηση (remapping), ώστε να μην υπάρχουν «κενά». Οπότε στο συγκεκριμένο παράδειγμα, η αρίθμηση θα εκτεινόταν από το 1 εώς το 48. Επιπλέον, στον πρώτο τομέα του qtop, Λίστα 2.7 στη γραμμή 3 παρουσιάζεται ο πραγματικός αριθμός των κόμβων, κι επιπλέον αναφέρεται πόσοι από αυτούς είναι λειτουργικοί. Οι 6 κόμβοι που λείπουν (42/48) μπορούν να αναζητηθούν στις στήλες του πίνακα όπου υπάρχει το γράμμα 'd' (down, μη-λειτουργικός).

Το νόημα των υπόλοιπων γραμμάτων της γραμμής 5 εξηγείται στην αρχή της ενότητας. Στις γραμμές που ακολουθούν (γραμμές 5-12) υπάρχει μια αλληλουχία αριθμών. Ο κάθε αριθμός από αυτούς είναι το αναγνωριστικό ενός κόμβου (Node ID), το οποίο συνδέεται με ένα λογαριασμό unix στον τρίτο τομέα του qtop (βλ. λίστα 2.9). Για παράδειγμα ο αριθμός 0 αντιστοιχεί στο λογαριασμό χρήστη prdatl28.

Τέλος, οι διέσεις (#) συμβολίζουν την απουσία πυρήνων στα μηχανήματα. Στο συγκεκριμένο παράδειγμα, οι κόμβοι 3 εώς και 19 έχουν μόνο τέσσερεις πυρήνες, και όχι οκτώ, όπως οι υπόλοιποι κόμβοι.

Επίσης, πρέπει να σημειωθεί ότι η έξοδος του qtop είναι κανονικά έγχρωμη. Κάθε λογαριασμός unix, και κατά συνέπεια κάθε αναγνωριστικό κόμβου στον πίνακα αντιστοιχεί σε ένα διαφορετικό χρώμα (ή μια ομάδα λογαριασμών μπορεί να περιγράφεται από το ίδιο χρώμα), γεγονός που κάνει τον πίνακα πολύ πιο ευανάγνωστο.

Πολλαπλότητα πινάκων

Ο πίνακας με την πληρότητα των κόμβων δεν χρειάζεται να είναι μόνο ένας. Ο αριθμός των πινάκων αυξάνεται ανάλογα με τον αριθμό των κόμβων μιας συστοιχίας, και εξαρτάται επιπλέον από το εύρος του παραθύρου στο οποίο απεικονίζεται. Αν για παράδειγμα το τερματικό μας χωράει 100 χαρακτήρες, και οι κόμβοι της συστοιχίας είναι 150, τότε αναγκαστικά θα δημιουργηθεί κι ένας δεύτερος πίνακας ακριβώς από κάτω.

Το εύρος του κάθε πίνακα μπορεί επίσης να καθοριστεί από το χρήστη (βλ. υποενότητα 3.2.1), ανεξάρτητα από τον αριθμό των κόμβων.

2.5 Τρόποι εκτέλεσης

Όπως είδαμε στην ενότητα 2.2, σελίδα 15, το qtop διαβάζει την έξοδο των εντολών `pbsnodes -a`, `qstat`, και `qstat -q`. Οι εντολές αυτές πρέπει να εκτελεστούν τοπικά στο Computing Element του εκάστοτε cluster. Αν έχουμε απευθείας πρόσβαση σε ένα τέτοιο μηχάνημα, τότε τρέχουμε εκεί το qtop, το qtop ζητάει μόνο του την εκτέλεση των παραπάνω εντολών και παίρνει την έξοδό τους. Αν η συστοιχία που μας ενδιαφέρει είναι μέρος του Grid, τότε μπορούμε να στείλουμε το qtop μέσα σε ένα jdl αρχείο σε ένα ή περισσότερα Computing Elements, αυτό να εκτελεστεί απομακρυσμένα σε καθένα από τα Computing Elements και τα αποτελέσματα της εκτέλεσής του να μας σταλούν πίσω. Ένας εναλλακτικός χειρισμός[6], που είναι και αυτός που προτιμήσαμε για τις ανάγκες της παρούσας διπλωματικής εργασίας[7], είναι να ζητήσουμε την εκτέλεση των τριών παραπάνω εντολών απομακρυσμένα (δηλαδή η εκτέλεσή τους να είναι η εργασία που ζητάμε) και να τροφοδοτήσουμε το επιστρεφόμενο αποτέλεσμα στο qtop, τοπικά πλέον στο UI που μας εξυπηρετεί και αφού επιστρέψει το αποτέλεσμα της εκτέλεσής τους (αποθηκευμένη σε τρία αρχεία, `pbsnodes_a`, `qstat.txt` και `qstat_q.txt` αντίστοιχα).

[6] Προφανώς θα ήταν αδύνατο να έχουμε τοπική πρόσβαση στα Computing Element όλων των διαθέσιμων σε μας συστοιχιών του Grid!

[7] Η διαδικασία περιγράφεται αναλυτικά στο εδάφιο 1.6.1

Κεφάλαιο 3

Ο κώδικας

Σημείωση: Η τεκμηρίωση του κώδικα αντιστοιχεί στο στιγμιότυπο του κώδικα `ab52fad2907f490d989ff5d4851bd25cf6a0ca85` *και μπορεί να βρεθεί στον τόπο:* `https://github.com/sfranky/qtop`.

3.1 Δομή

Το qtop αποτελείται από τα ακόλουθα τρία αρχεία:

- **qtop.py**, που είναι το κύριο script που εκτελείται,
- **qtop.conf**, που είναι το αρχείο ρυθμίσεων,
- **qtop.colormap**, που περιέχει την πληροφορία χρώματος για την έγχρωμη απεικόνιση του πίνακα και των δεδομένων.

3.2 qtop.py

Στις γραμμές 71–77 εισάγονται μερικές βιβλιοθήκες που χρειάζονται αργότερα για την εκτέλεση του προγράμματος.
Στις γραμμές 79–87 δημιουργούνται τα ορίσματα της γραμμής εντολών (Command-line arguments, βλ. υποενότητα 3.2.1).
Στις γραμμές 138–650 ορίζονται οι συναρτήσεις του προγράμματος (βλ. υποενότητα 3.2.2).

Κύριο πρόγραμμα

Στη γραμμή 667, όπου ουσιαστικά ξεκινάει το πρόγραμμα, διαβάζεται το αρχείο ρυθμίσεων, `qtop.conf`.
Στις γραμμές 669–714 καλούνται οι συναρτήσεις που διαβάζουν την έξοδο των εντολών `pbsnodes -a`, `qstat`, και `qstat -q`, και δημιουργούν τα αρχεία YAML.

Στη συνέχεια μέχρι τη γραμμή 721 καθορίζονται κάποιες μεταβλητές που σχετίζονται με το διαθέσιμο χώρο στο τερματικό της οθόνης, για την απεικόνιση του qtop. Πιο αναλυτικά, η `os.popen('stty size', 'r').read().split()` επιστρέφει το εύρος σε γραμμές και στήλες του τερματικού στο οποίο τρέχει το qtop, και η `DEADWEIGHT` είναι το εύρος του κειμένου που ακολουθεί τη γραμμή κατάστασης του κάθε πυρήνα (δηλαδή της έκφρασης «`=CoreXX`»).
Στη γραμμή 723 τυπώνεται ο πρώτος τομέας της εξόδου του qtop.
Έπειτα, στις γραμμές 726–750 γίνεται μια καταμέτρηση των εργασιών του κάθε χρήστη ανά κατάσταση, οι οποίες έπειτα εκχωρούνται σε κατάλληλα λεξικά, ενώ όπου δεν υπάρχει καταχώρηση η αντίστοιχη θέση μηδενίζεται. Στο λεξικό `Usersortedlst` καταχωρούνται στη σειρά οι χρήστες με βάση τη συχνότητα εμφάνισής τους.
Γραμμές 753–765, αν ο αριθμός χρηστών υπερβεί τα διαθέσιμα σύμβολα που τους αντιπροσωπεύουν, χρησιμοποιείται διψήφιο σύμβολο (μη λειτουργική λύση, πρέπει να επιλυθεί αποτελεσματικότερα στο μέλλον) και έπειτα γίνεται η αντιστοίχιση χρήστη–αναγνωριστικού.
Οι γραμμές 766–786 υπολογίζουν και ταξινομούν με το μέγεθος τις γραμμές με τις αντιστοιχίσεις χρηστών–αναγνωριστικών, τις εργασίες που τρέχουν/είναι σε αναμονή και το άθροισμά τους.
Στις γραμμές 787–800 δημιουργείται ένα λεξικό για κάθε γραμμή–πυρήνα του πίνακα του qtop και εκτελείται η `fill_cpucore_columns` ανάλογα με το αν έχει επιλεγεί (ή επιβάλλεται) αναδιάταξη αρίθμησης.

Ακολουθεί ο υπολογισμός και η εκτύπωση του δεύτερου τομέα του qtop. Ξανά, ανάλογα με το αν έχει επιλεγεί αναδιάταξη αρίθμησης:

- εκτελείται η `calculate_Total_WNIDLine_Width` με το κατάλληλο όρισμα,
- υπολογίζεται η γραμμή NodeState,
- εκτελείται η `print_WN_ID_lines` με τα κατάλληλα ορίσματα.

Τελικά εκτυπώνεται η γραμμή NodeState.
Στη συνέχεια, γραμμές 829–833, συμπληρώνεται το λεξικό AccountNrlessOfId ανάλογα με τα περιεχόμενα του λεξικού AccountsMappings.
Στις γραμμές 839–849 γίνεται ο χρωματισμός της γραμμής κάθε πυρήνα, αφού αυτή μετατραπεί προσωρινά σε λίστα, και μέσω της κλήσης της συνάρτησης `Colorize()`.

3.2.1 Ορίσματα γραμμής εντολών

- `"-a", "--blindremapping"`

  Επιβάλλει την αναδιάταξη αρίθμησης ανεξάρτητα από την πληρότητα πίνακα, τους ανύπαρκτους κόμβους στην αρχή του πίνακα ή την ανομοιογένεια των κόμβων.

- `"-c", "--COLOR"`

  Επιτρέπει την επιλογή έγχρωμης ή μονόχρωμης εξόδου.

- `"-f", "--setCOLORMAPFILE"`

  Επιτρέπει τον ορισμό εναλλακτικού αρχείου χρωμάτων για τους χρήστες που έχουν αναθέσει εργασίες στο cluster.

- `"-m", "--noMasking"`

- `"-o", "--SetVerticalSeparatorXX"`

  Ορίζει μια κατακόρυφη στήλη κάθε «XX» στήλες στον πίνακα του qtop. Η προκαθορισμένη ρύθμιση «0» δεν προσθέτει καμμία στήλη.

- `"-s", "--SetSourceDir"`

  Ορίζει τον κατάλογο που βρίσκονται οι έξοδοι των εντολών `pbsnodes -a`, `qstat`, και `qstat -q`.

- `"-F", "--ForceNames"`

  Εξαναγκάζει τη χρήση ονομάτων αντί για αριθμούς στον πίνακα του qtop (στην περίπτωση που οι κόμβοι δεν είναι ονομασμένοι με αριθμούς, αλλά αποκλειστικά με ονόματα.

3.2.2 Συναρτήσεις

`make_pbsnodes_yaml()`

Η συνάρτηση `make_pbsnodes_yaml()` διαβάζει το αρχείο-έξοδο της `pbsnodes -a` γραμμή-γραμμή και συγκεντρώνει στοιχεία, τα οποία γράφει σε ένα νέο αρχείο, σε YAML format. Τα στοιχεία αυτά περιγράφονται στην ενότητα 2.2.1. Επιπλέον, αν το πρόγραμμα διαγνώσει ότι έχει καταχωρηθεί παραπάνω από μία εργασία στον ίδιο πυρήνα, βγάζει μήνυμα σφάλματος και διακόπτει την εκτέλεση του.

`read_pbs_nodes_yaml(fin, fout)`

Η `read_pbs_nodes_yaml(fin, out)` διαβάζει το ενδιάμεσο αρχείο που δημιουργήθηκε από την προηγούμενη συνάρτηση και αρχίσει να συλλέγει τις απαραίτητες πληροφορίες: Για την αποθήκευση του domain name, χρησιμοποιούνται Regular Expressions[1]. Το πρόθεμα (ή τα προθέματα και τα επιθέματα) και η αρίθμηση του domain name αποθηκεύονται ξεχωριστά, ώστε

- σε περίπτωση ανομοιογένειας μιας συστοιχίας, να εφαρμόζεται αναδιάταξη της αρίθμησης (remapping), ή, μελλοντικά, οι διαφορετικές υπο-συστοιχίες να μπορούν να απεικονιστούν σε διαφορετικούς πίνακες. Ανομοιογένεια παρουσιάζεται όταν π.χ. οι κόμβοι αρχικά ονομάζονται wn003, wn004, wn005

[1]Τα Regular Expressions «παρέχουν ένα συνοπτικό και ευέλικτο τρόπο (προσδιορισμού και αναγνώρισης) συμβολοσειρών κειμένου, όπως είναι οι συγκεκριμένοι χαρακτήρες, οι λέξεις, ή τα μοτίβα χαρακτήρων. [14]»

κ.λπ. και μετά από λίγο η ονομασία αλλάζει π.χ. σε gn004, gn005, gn006 κ.λπ.

- στην περίπτωση έλλειψης αριθμών στην ονομασία των κόμβων (αν οι κόμβοι είναι ολιγάριθμοι κι έχουν διακριτά ονόματα, όπως π.χ. ονόματα επιστημόνων, ή τόπων, ή φρούτων) να εφαρμόζεται αναδιάταξη της αρίθμησης.
- σε όλες τις άλλες περιπτώσεις η αρίθμηση να αντιστοιχεί στη δοσμένη από το διαχειριστή.

Για την αποθήκευση της κατάστασης του κόμβου, ακολουθείται η λογική που παρουσιάστηκε στην ενότητα 2.4.

Επιπλέον, αποθηκεύονται ο αριθμός πυρήνων του κάθε κόμβου, ο μεγαλύτερος κόμβος που εκτελεί εργασία και το αναγνωριστικό της εργασίας κάθε κόμβου. Οι θέσεις των ανύπαρκτων κόμβων καλύπτονται με ένα ερωτηματικό, «?» για χρήση αργότερα στη γραμμή Node State του πίνακα.

Στο τέλος της συνάρτησης γίνεται χειρισμός των περιπτώσεων (α) η δοσμένη αρίθμηση των κόμβων να ξεκινάει από πολύ μεγάλο αριθμό (> 9000) και (β) ο πίνακας να είναι αραιοκατοικημένος. Και στις δύο περιπτώσεις εκτελείται αναδιάταξη των κόμβων.

make_qstatq_yaml(fin, fout)

Η `make_qstatq_yaml(fin, fout)` διαβάζει την έξοδο του `qstat_q.txt` και συλλέγει τα ονόματα των ουρών, πόσες εργασίες τρέχουν, πόσες εργασίες είναι σε αναμονή, ποιο είναι το όριο σε εργασίες που μπορούν να εκτελεστούν στην ουρά ταυτόχρονα και τέλος ποια είναι η κατάσταση της ουράς (Για αναλυτικότερη παρουσίαση των στοιχείων αυτών, βλ. υποενότητα 2.2.3). Οι πληροφορίες αυτές, μαζί με το σύνολο τρέχουσων και σε αναμονή εργασιών, αποθηκεύονται στο αρχείο `qstat_q.yaml`.

make_qstat_yaml(fin, fout)

Η `make_qstat_yaml(fin, fout)` διαβάζει την έξοδο του `qstat.txt` και συλλέγει το αναγνωριστικό της εργασίας, το όνομα του χρήστη, την κατάσταση της ουράς, και το όνομα της ουράς, τα οποία και γράφει στο αντίστοιχο αρχείο `qstat.yaml` . Η συνάρτηση προς το παρόν μπορεί να αντιμετωπίσει δύο διαφορετικές μορφές παρουσίασης του αρχείου `qstat.txt`. Η εναλλακτική παρουσίαση έχει τη μορφή:

```
1 job-ID prior name user state submit/start queue slots ja-task-ID
```

Οι συλλεχθείσες πληροφορίες αποθηκεύονται στο αρχείο `qstat.yaml`

read_qstat()

Η `read_qstat()` διαβάζει το αρχείο `qstat.yaml` και προσθέτει τις αντίστοιχες πληροφορίες σε κατάλληλες λίστες.

job_accounting_summary()

Η `job_accounting_summary()` τυπώνει τον πρώτο τομέα της εξόδου του qtop.

fill_cpucore_columns(value, CPUDict)

Η `fill_cpucore_columns()` υπολογίζει και συμπληρώνει την κατάσταση εργασίας για όλους τους πυρήνες κάθε κόμβου. Η πληροφορία αποθηκεύεται σε λεξικά (dictionaries), όπου κάθε λεξικό αντιστοιχεί σε έναν πυρήνα, και περιέχει την κατάσταση του αντίστοιχου πυρήνα για *όλους* τους κόμβους της συστοιχίας. Για παράδειγμα, στη λίστα 2.8, σελ. 22, κάθεμία από τις γραμμές 5–12 είναι και ένα λεξικό στο οποίο είναι αποθηκευμένη μια συμβολοσειρά κειμένου, όπου κάθε χαρακτήρας αντιστοιχεί μέσω ενός αναγνωριστικού σε ένα λογαριασμό χρήστη.

Όταν ένας πυρήνας είναι ελεύθερος από εργασίες, η αντίστοιχη θέση καλύπτεται με το σύμβολο «_». Όταν σε έναν κόμβο υπάρχουν λιγότεροι πυρήνες απ'ο,τι σε άλλους κόμβους με περισσότερους πυρήνες, η αντίστοιχη θέση καλύπτεται με το σύμβολο «#».

Η συνάρτηση κάνει έλεγχο για την έγκυρη αντιστοίχιση του κάθε αναγνωριστικού εργασίας (jobID) με ένα χρήστη. Αν κάποιο αναγνωριστικό δεν υπάρχει, το πρόγραμμα τερματίζεται δίνοντας στην έξοδο κατάλληλο μήνυμα σφάλματος.

Επιπλέον, το ύψος του πίνακα καθορίζεται από τον μεγαλύτερο διαθέσιμο πυρήνα. Αν οι πρώτοι 100 κόμβοι έχουν 16 πυρήνες, αλλά οι υπόλοιποι 200 κόμβοι έχουν μόνο 8 πυρήνες, τότε οι επιπλέον πίνακες που σχεδιάζονται θα έχουν ύψος μόνο 8 πυρήνων και όχι 16.

insert(original, new, pos, stopaftern = 0)

Η συνάρτηση `insert(original, new, pos, stopaftern = 0)` εισάγει κατακόρυφο διαχωριστικό ανά x θέσεις, που καθορίζονται από το χρήστη, μέσω του ορίσματος της γραμμής εντολών `-o` (βλ. 3.2.1). Υπάρχει προαιρετική μεταβλητή η οποία καθορίζει μετά από πόσες επαναλήψεις σταματάει η εισαγωγή διαχωριστικών. Στην προκαθορισμένη επιλογή, δεν εισάγονται διαχωριστικά.

calculate_Total_WNIDLine_Width(WNnumber)

Η `calculate_Total_WNIDLine_Width(WNnumber)` υπολογίζει την κατακόρυφη αρίθμηση των κόμβων (για παράδειγμα, στις γραμμές 2–3 της λίστας 2.8). Στην ουσία παράγει συμβολοσειρές κειμένου κατάλληλου μήκους ανάλογα με τον αριθμό των κόμβων, που περνάει στη συνάρτηση ως παράμετρος.

find_Matrices_Width(WNnumber, WNList)

Η `find_Matrices_Width(WNnumber, WNList)` επιτελεί τα εξής:

- Αν η αρίθμηση των κόμβων ξεκινάει μετά το 1, για παράδειγμα ξεκινάει από το 15, κρύβει τους ανύπαρκτους κόμβους 1–14, ώστε να μην εμφανίζονται στήλες με ερωτηματικά στον πίνακα. (masking/clipping)

- Υπολογίζει τον αριθμό των επιπλέον πινάκων που μπορεί να χρειαστούν για την απεικόνιση του qtop.

- Αν ο χρήστης έχει δώσει χειροκίνητα μια τιμή στη μεταβλητή `UserCutMatrixWidth` στο αρχείο `qtop.conf`, το μέγιστο εύρος του πίνακα καθορίζεται από τη μεταβλητή αυτή, με αποτέλεσμα να δημιουργούνται επιπλέον πίνακες για να χωρέσουν όλοι οι κόμβοι.

`print_WN_ID_lines(start, stop, WNnumber)`

Η `print_WN_ID_lines(start, stop, WNnumber)` τυπώνει τις γραμμές που αφορούν την αρίθμηση των κόμβων του πίνακα (βλ. γραμμές 2–3 στη λίστα 2.8), ανάλογα με τον αριθμό κόμβων, και ανάλογα με το αν υπάρχουν κατακόρυφες διαχωριστικές γραμμές. Επίσης, αν δεν υπάρχει αρίθμηση στους κόμβους, τυπώνει τα ονόματα των κόμβων κατακόρυφα και με εναλλάξ χρωματισμό, ώστε να είναι ευδιάκριτα και ευανάγνωστα.

`reset_yaml_files()`

Η `reset_yaml_files()` σβήνει τα αρχεία yaml που δημιουργούνται κατά το τρέξιμο του qtop, στην περίπτωση που έχει επιλεγεί να γράφονται στον ίδιο κατάλογο κάθε φορά.

3.3 qtop.conf

Το αρχείο αυτό είναι το αρχείο ρυθμίσεων. Εδώ ρυθμίζονται παράμετροι και σταθερές.

Η σταθερά `MIN_MASKING_THRESHOLD` ορίζει τον ελάχιστο αριθμό από τον οποίο και πάνω ενεργοποιείται αυτόματα η λειτουργία αναδιάταξης αρίθμησης (remapping) των κόμβων. Για παράδειγμα, για την προκαθορισμένη τιμή «9», αν ο πρώτος Worker Node ξεκινάει με τον αριθμό 10, τότε οι πρώτοι 9 αριθμοί θα παραλειφθούν από τον πίνακα του qtop και η αρίθμηση θα αναδιαταχτεί ώστε οι κόμβοι να είναι πλέον συνεχόμενοι. Αν ο πρώτος Worker Node ξεκινάει με τον αριθμό 5, τότε θα εμφανιστούν κανονικά οι πρώτοι τέσσερεις αριθμοί με ερωτηματικά στις αντίστοιχες στήλες.

Η `UserCutMatrixWidth` ορίζει χειροκίνητα το εύρος του κάθε πίνακα του qtop. Η προκαθορισμένη ρύθμιση «0» αφήνει το εύρος του πίνακα να προσδιοριστεί από το εύρος του παραθύρου του τερματικού στο οποίο απεικονίζεται το qtop.

Η SEPARATOR ορίζει το είδος της κατακόρυφης στήλης. Προκαθορισμένη είναι μια απλή κατακόρυφος, («|»).

Η PERCENTAGE είναι το ποσοστό «πληρότητας» του πίνακα, από το οποίο και κάτω ενεργοποιείται η λειτουργία αναδιάταξης αρίθμησης των κόμβων. Αν δηλαδή ένας πίνακας αριθμεί 100 κόμβους αλλά μόνο οι 70 έχουν εργασίες, οι υπόλοιποι 30 κόμβοι δεν απεικονίζονται, και η αρίθμηση είναι συνεχόμενη από το 1 εώς το 70.

Η POSSIBLE_IDS είναι μια λίστα που περιέχει όλα τα δυνατά αναγνωριστικά λογαριασμών unix που αντιστοιχίζονται στους χρήστες που έχουν στείλει εργασίες. Περιλαμβάνονται τα 26 γράμματα του αγγλικού αλφαβήτου, κεφαλαία και μικρά (σύνολο 52), και οι αριθμοί 0 εώς 9. Το αναγνωριστικό πρέπει αναγκαστικά να είναι μονοψήφιο ώστε να καταλαμβάνει μία μόνο θέση στον πίνακα.

Έπειτα, στις γραμμές 16-24 ορίζονται μερικοί κατάλογοι και ονομασίες αρχείων.

Στις γραμμές 31-33 ανιχνεύονται τα αρχεία-έξοδοι των εντολών `pbsnodes- a`, `qstat` και `qstat -q`.

3.4 qtop.colormap

Το αρχείο αυτό περιέχει δύο λεξικά, το ColorOfAccount και το CodeOfColor. Το πρώτο αντιστοιχίζει κάθε λογαριασμό χρήστη (μόνο το μέρος που περιέχει γράμματα, χωρίς τους αριθμούς) σε ένα χρώμα, και το δεύτερο αντιστοιχίζει το χρώμα στον κωδικό ansi του[2]. Στο λεξικό ColorOfAccount, για ευκολία έχουν προστεθεί και οι οδηγίες για το χρωματισμό των στοιχείων «_», «#», και των λογαριασμών που δεν έχουν χρώμα («NoColourAccount»).

Είναι δυνατό στο χρήστη να παρέχει δικό του `qtop.colormap` αρχείο με λεξικά με διαφορετικές οδηγίες για το χρωματισμό των λογαριασμών.

[2] βλ. http://en.wikipedia.org/wiki/ANSI_escape_code

3.5 Συμπεράσματα – Παρατηρήσεις

Όπως προαναφέρθηκε, ο κύριος λόγος που το qtop ξαναγράφτηκε από την αρχή ήταν η ανάγκη για επέκταση και η ανάγκη για εξάλειψη των bugs που το καθιστούσαν μη λειτουργικό σε αρκετά συστήματα που παρουσιάζουν μια κάποια πολυπλοκότητα, ή ιδιαιτερότητες που δεν μπορούσαν να είναι γνωστές 3 χρόνια πριν. Καλύφθηκαν αυτές οι ανάγκες; Η απάντηση είναι πως ναι. Τα αποτέλεσματα από την εκτέλεση του qtop που παρουσιάζονται στο παράρτημα Β (διαθέσιμα μόνο στην ηλεκτρονική έκδοση της εργασίας), σχεδόν στο σύνολό τους δεν μπορούν να αναπαραχθούν από το αρχικό qtop. Το νέο qtop είναι σε θέση να απεικονίσει και να συνοψίσει συστοιχίες αρκετά πιο πολύπλοκες (με μεγαλύτερο βαθμό ανομοιογένειας), αλλά και συστοιχίες που παλιότερα απλά αδυνατούσε να απεικονίσει λόγω bugs. Ο μελλοντικός χειρισμός διαφορετικών LRMS (Local Resource Management Systems) είναι πιο εύκολος, εφόσον είναι πλέον θέμα να γραφτεί η κατάλληλη επέκταση. Ένας σημαντικός τομέας στον οποίο βελτιώθηκε το qtop είναι και αυτός της ταχύτητας: σε πολύπλοκες συστοιχίες, το qtop εκτελείται αισθητά γρηγορότερα, και αναμένεται στο μέλλον να βελτιωθεί περαιτέρω, αφού ο κώδικας ακόμα είναι «νωπός», και επιδέχεται σίγουρα βελτιώσεις και βελτιστοποιήσεις.

3.5.1 Χρησιμότητα του qtop

Οι πληροφορίες που το qtop αξιοποιεί δεν παράγονται από το ίδιο το qtop, αλλά προέρχονται από την έξοδο τριών εντολών, `pbsnodes -a`, `qstat`, και `qstat -q`, που βρίσκονται σε κάθε Computing Element και εν γένει queueing system. Μια προφανής ερώτηση που είναι φυσικό να έρθει στο νου είναι, εφόσον το qtop δεν παράγει καινούργια πληροφορία, πού έγκειται η χρησιμότητα του; Με άλλα λόγια, τι είναι αυτό που «φέρνει στο τραπέζι»;

Αυτό που δεν είναι τόσο προφανές στον αναγνώστη είναι ότι η έξοδος καθεμίας από αυτές τις εντολές μπορεί να είναι οσοδήποτε μεγάλη–και στα σύγχρονα συστήματα είναι όντως έτσι. Η έξοδος του `pbsnodes -a` μπορεί να είναι δέκα γραμμές, αλλά μπορεί να είναι και δεκάδες χιλιάδες γραμμές, στην οποία περίπτωση ο έλεγχός της από τον διαχειριστή της εκάστοτε συστοιχίας είναι εξαιρετικά χρονοβόρος και επιρρεπής σε λάθη. Το qtop συλλέγει την πληροφορία και την παρουσιάζει με ένα τρόπο εποπτικό, σε μια εικόνα, όπου ο διαχειριστής μπορεί με μια ματιά να δει απευθείας ποια είναι η κατάσταση ενός κόμβου ή που εντοπίζεται το πρόβλημα.

Ένα σενάριο όπου το qtop φαίνεται ιδιαίτερα χρήσιμο είναι όταν κάτι πάει στραβά. Για παράδειγμα, μπορεί ένας σκληρός δίσκος να αστοχήσει και να τεθεί εκτός λειτουργίας, ή απλά να μην είναι προσβάσιμος. Αν η εργασία εκείνη την ώρα εκτελείται στη μνήμη, όταν περατωθεί και δε μπορεί να αποθηκευτεί στο δίσκο, είναι πολύ πιθανό το σύστημα να μη σταματήσει ποτέ να την παρουσιάζει «σε εξέλιξη». Ο διαχειριστής μπορεί να δει ποιος χρήστης επηρεάζεται και να τον ειδοποιήσει για την κατάσταση των εργασιών του.

Ή, σε ένα αντίστροφο παράδειγμα, κατά τη διάρκεια ενός security challenge,

μπορεί ένας «χρήστης–hacker» να μπει στο σύστημα. Το qtop βοηθάει να βρεθούν άμεσα οι εργασίες που ο hacker έχει στείλει, ώστε ο διαχειριστής να λάβει τα απαραίτητα μέτρα προστασίας του συστήματος.

Το qtop, τέλος, είναι ιδιαίτερα χρήσιμο και παραστατικό στην περίπτωση που ο χρήστης(/διαχειριστής) μπορεί να το τρέξει τοπικά σε μια συστοιχία. Με τη χρήση της εντολής `watch -d` αμέσως πριν την κλήση του qtop, η έξοδος του qtop ανανεώνεται κάθε 2 δευτερόλεπτα, δείχνοντας ταυτόχρονα τα σημεία όπου υπήρξε αλλαγή, με αποτέλεσμα ο χρήστης να έχει μια σχεδόν real-time εποπτική εικόνα της κατάστασης της συστοιχίας, και να μπορεί να ελέγξει την ορθή λειτουργία του συστήματος παρακολουθώντας όλη τη δυναμική ροή εργασιών, ακριβώς τη στιγμή που αυτό συμβαίνει.

3.5.2 Μια χρήσιμη εφαρμογή

Το qtop μπορεί να σταλεί για εκτέλεση και συλλογή πληροφοριών σε όλες τις συστοιχίες όλων των VO στα οποία είναι εγεγραμμένος ένας χρήστης, κατά κάποιο τρόπο παρόμοια με τους web crawlers των μηχανών αναζήτησης (google, yahoo κ.λπ.) που, σαν αράχνες, περνάνε από όλα τα σημεία του παγκόσμιου ιστού και καταγράφουν την κατάστασή του και τα περιεχόμενά του. Επειδή ακριβώς είναι τόσο συνοπτική και παραστατική η απεικόνισή του, και αν ο χρήστης είναι εγεγραμμένος σε «σημαντικά» (μεγάλα σε μέγεθος) VO, η συγκέντρωση αυτής της πληροφορίας μπορεί να παρέχει μια πάρα πολύ αναλυτική αποτύπωση της κατάστασης των VO. Το σύνολο αυτής της πληροφορίας παρέχει δηλαδή ένα μερικό στιγμιότυπο, ένα αποτύπωμα ενός μεγάλου κομματιού του Grid, επιτρέποντας έτσι να βγάλει κανείς χρήσιμα συμπεράσματα και στατιστικές για τη δομή και σύσταση του Grid ανά πάσα στιγμή, ή σε βάθος χρόνου. Στην παρούσα εργασία έγιναν τα πρώτα βήματα για να πραγματοποιηθεί μία τέτοια μελέτη, δηλαδή συλλέχθηκαν τα αντίστοιχα στοιχεία, και μάλιστα με σχετική ευκολία. Ίσως μελλοντικά αξίζει να πραγματοποιηθεί μια τέτοια μελέτη επισταμένα.

Παράρτημα Α΄

Πηγαίος Κώδικας

Η επιλογή της γλώσσας προγραμματισμού του qtop

Το πρόγραμμα γράφτηκε στην προγραμματιστική γλώσσα python. Οι λόγοι γι' αυτή την επιλογή είναι αρκετοί, αλλά οι κύριοι είναι ότι:

- είναι interpreted γλώσσα προγραμματισμού, το οποίο καθιστά την ανάπτυξη κώδικα σ' αυτή πολύ πιο γρήγορη απ' ό,τι στις μεταγλωτισμένες (compiled) γλώσσες,
- είναι δωρέαν, τρέχει σε ένα ευρύ φάσμα υπολογιστικών συστημάτων, και βρίσκεται ήδη εγκατεστημένη σε όλα τα User Interface,
- οι πολυάριθμες έτοιμες βιβλιοθήκες την καθιστούν ιδιαίτερα δυνατή για τέτοιου είδους εργασίες.

Επίσης, το πρόγραμμα έχει γραφτεί με τρόπο τέτοιο που να διατηρείται η συμβατότητα με τις πιο παλιές εκδόσεις της python (από έκδοση 2.4 μέχρι και 3.1) που βρίσκονται ακόμα εγκατεστημένες σε πολλά μηχανήματα του Grid.

Στο παράρτημα αυτό δίνεται ο πηγαίος κώδικας της εφαρμογής qtop, μαζί με τα αρχεία ρυθμίσεών του, qtop.conf και qtop.colormap. Οι τελευταίες εκδόσεις του πηγαίου κώδικα βρίσκονται στη διεύθυνση: `https://github.com/sfranky/qtop`.

qtop.conf

```
#!/usr/bin/env python
# Masking/clipping functionality:
# How high should the earliest node number be
# (e.g. 50), to make the previous WNs vanish
MIN_MASKING_THRESHOLD = 9
```

```
# Change this to zero if you don't want a custom cut matrix length
UserCutMatrixWidth = 0
#type of vertical separator for -o switch
SEPARATOR = '|'
# if less than PERCENTAGE% of the matrices have jobs, perform a blind remap
PERCENTAGE = 0.8
# IDs of unix accounts, for the lower part of qtop
# POSSIBLE_IDS = '0123456789 ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz'
POSSIBLE_IDS = ['0','1','2','3','4','5','6','7','8','9','A','B','C','D','E','F','G','H','I','J','K','L','M','N','O','P','Q','R','S','T','U','V','W','X','Y','Z','a','b','c','d','e','f','g','h','i','j','k','l','m','n','o','p','q','r','s','t','u','v','w','x','y','z']

HOMEPATH = os.path.expanduser(' /')
QTOPPATH = os.path.expanduser(' /qtop/qtop')
PROGDIR = os.path.expanduser(' /off/qtop')
SOURCEDIR = options.SOURCEDIR # as set by the '-s' switch

# the following three lines save the produced YAML files in the dataset folder each time
PBSNODES_YAML_FILE = 'pbsnodes.yaml'
QSTATQ_YAML_FILE = 'qstat-q.yaml'
QSTAT_YAML_FILE = 'qstat.yaml'

dir = SOURCEDIR

os.chdir(dir)

# Location of read and created files
PBSNODES_ORIG_FILE = [f for f in os.listdir(os.getcwd()) if f.startswith('pbsnodes') and not f.endswith('.yaml')][0]
QSTATQ_ORIG_FILE = [f for f in os.listdir(os.getcwd()) if (f.startswith('qstat_q') or f.startswith('qstatq') or f.startswith('qstat-q') and not f.endswith('.yaml'))][0]
QSTAT_ORIG_FILE = [f for f in os.listdir(os.getcwd()) if f.startswith('qstat.') and not f.endswith('.yaml')][0]
#PBSNODES_ORIG_FILE = 'pbsnodes.out'
#QSTATQ_ORIG_FILE = 'qstat-q.out'
#QSTAT_ORIG_FILE = 'qstat.out'
```

qtop.conf

qtop.py

```
#!/usr/bin/env python

################################################
#              qtop v.0.6.4                    #
#     Licensed under MIT-GPL licenses          #
#                     Fotis Georgatos          #
#                     Sotiris Fragkiskos       #
################################################

"""
changelog:
=========
0.6.4: lines that don't contain *any* actual core are now not
   printed in the matrices.
0.6.3: optional stopping of vertical separators (every 'n'
   position for x times)
       additional vertical separator in the beginning
0.6.2: WN matrix width bug ironed out.
0.6.1: Custom-cut matrices (horizontally, too!), -o switch
0.5.2: Custom-cut matrices (vertically, not horizontally), width
    set by user.
0.5.1: If more than 20% of the WNs are empty, perform a blind
   remap.
       Code Cleanup
0.5.0: Major rewrite of matrices calculation
       fixed: true blind remapping !!
       exotic cases of very high numbering schemes now handled
       more qstat entries successfully parsed
       case of many unix accounts (>62) now handled
0.4.1: now understands additional probable names for pbsnodes,
   qstat and qstat-q data files
0.4.0: corrected colorless switch to have ON/OFF option (default
    ON)
       bugfixes (qstat_q didn't recognize some faulty cpu time
           entries)
       now descriptions are in white, as before.
       Queues in the job accounting summary section are now
           coloured
0.3.0: command-line arguments (mostly empty for now)!
       non-numbered WNs can now be displayed instead of numbered
            WN IDs
       fixed issue with single named WN
       better regex pattern and algorithm for catching
           complicated numbered WN domain names
       implement colorless switch (-c)
0.2.9: handles cases of non-numbered WNs (e.g. fruit names)
```

```
37        parses more complex domain names (with more than one dash
              )
38        correction in WN ID numbers display (tens were
              problematic for larger numbers)
39 0.2.8: colour implementation for all of the tables
40 0.2.7: Exiting when there are two jobs on the same core reported
       on pbsnodes (remapping functionality to be added)
41        Number of WNs >1000 is now handled
42 0.2.6: fixed some names not being detected (%,= chars missing
      from regex)
43        changed name to qtop, introduced configuration file qtop.
              conf and
44        colormap file qtop.colormap
45 0.2.5: Working Cores added in Usage Totals
46        Feature added: map now splits into two if terminal width
              is smaller than
47         the Worker Node number
48 0.2.4: implemented some stuff from PEP8
49        un-hardwired the file paths
50        refactored code around CPUCoreDict functionality (
              responsible for drawing
51         the map)
52 0.2.3: corrected regex search pattern in make_qstat to recognize
       usernames like spec101u1 (number followed by number followed
       by letter) now handles non-uniform setups
53         R + Q / all: all did not display everything (E status)
54 0.2.2: masking/clipping functionality (when nodes start from e.g
      . wn101, empty columns 1-100 are ommited)
55 0.2.1: Hashes displaying when the node has less cores than the
      max declared by a WN (its np variable)
56 0.2.0: unix accounts are now correctly ordered
57 0.1.9: All CPU lines displaying correctly
58 0.1.8: unix account id assignment to CPU0, 1 implemented
59 0.1.7: ReadQstatQ function (write in yaml format using Pyyaml)
60        output up to Node state !
61 0.1.6: ReadPbsNodes function (write in yaml format using Pyyaml)
62 0.1.5: implemented saving to 3 separate files, QSTAT_ORIG_FILE,
      QSTATQ_ORIG_FILE, PBSNODES_ORIG_FILE
63 0.1.4: some "wiremelting" concerning the save directory
64 0.1.3: fixed tabs-to-spaces. Formatting should be correct now.
65        Now each state is saved in a separate file in a results
              folder
66 0.1.2: script reads qtop-input.out files from each job and
      displays status for each job
67 0.1.1: changed implementation in get_state()
68 0.1.0: just read a pbsnodes-a output file and gather the results
       in a single line
69 """
70
```

```
71 from operator import itemgetter
72 from optparse import OptionParser
73 import datetime
74 import glob
75 import os
76 import re
77 import sys
78
79 parser = OptionParser() # for more details see http://docs.
       python.org/library/optparse.html
80 parser.add_option("-a", "--blindremapping", action="store_true",
        dest="BLINDREMAP", default=False, help="This is used in
       situations where node names are not a pure arithmetic
       sequence (eg. rocks clusters)")
81 parser.add_option("-c", "--COLOR", action="store", dest="COLOR",
        default='ON', choices=['ON', 'OFF'], help="Enable/Disable
       color in qtop output. Use it with an ON/OFF switch: -c ON or
       -c OFF")
82 parser.add_option("-f", "--setCOLORMAPFILE", action="store",
       type="string", dest="COLORFILE")
83 parser.add_option("-m", "--noMasking", action="store_false",
       dest="MASKING", default=True, help="Don't mask early empty
       Worker Nodes. (default setting is: if e.g. the first 30 WNs
       are unused, counting starts from 31).")
84 parser.add_option("-o", "--SetVerticalSeparatorXX", action="
       store", dest="WN_COLON", default=0, help="Put vertical bar
       every WN_COLON nodes.")
85 parser.add_option("-s", "--SetSourceDir", dest="SOURCEDIR", help
       ="Set the source directory where pbsnodes and qstat reside")
86 parser.add_option("-z", "--quiet", action="store_false", dest="
       verbose", default=True, help="don't print status messages to
       stdout. Not doing anything at the moment.")
87 parser.add_option("-F", "--ForceNames", action="store_true",
       dest="FORCE_NAMES", default=False, help="force names to show
       up instead of numbered WNs even for very small numbers of WNs
       ")
88
89 (options, args) = parser.parse_args()
90
91 if not options.COLORFILE:
92     options.COLORFILE = os.path.expanduser(' /qtop/qtop/qtop.
          colormap')
93 qtopcolormap = open(options.COLORFILE, 'r')
94 exec qtopcolormap
95
96
97 def Colorize(text, pattern):
98     """print text colored according to its unix account colors
          """
```

```
99      if options.COLOR == 'ON':
100         return "\033[" + CodeOfColor[ColorOfAccount[pattern]] +
               "m" + text + "\033[1;m"
101     else:
102         return text
103
104 #for calculating the WN numbers
105 h1000, h0100, h0010, h0001 = '', '', '', ''
106 PrintStart, PrintEnd = 0, None
107
108 if options.FORCE_NAMES == False:
109     JUST_NAMES_FLAG = 0
110 else:
111     JUST_NAMES_FLAG = 1
112 RemapNr = 0
113 NodeSubClusters = set()
114 OutputDirs = []
115 HighestCoreBusy = 0
116 AllWNsDict, AllWNsRemappedDict = {}, {}
117 BiggestWrittenNode = 0
118 WNList, WNListRemapped = [], []
119 NodeNr = 0
120 NodeState = ''
121 ExistingNodes, OfflineDownNodes = 0, 0
122 MaxNP = 0
123 TotalCores, WorkingCores = 0, 0
124 TotalRuns, TotalQueues = 0, 0  # for readQstatQ
125 JobIds, UnixAccounts, Statuses, Queues = [], [], [], []  # for
       read_qstat
126 qstatqLst = []
127 UserOfJobId, IdOfUnixAccount = {}, {}
128 AccountsMappings = []
129 DIFFERENT_QSTAT_FORMAT_FLAG = 0
130
131 ### CPU lines ########################################
132
133 MaxNPRange = []
134
135 AccountNrlessOfId = {}
136 ####################################################
137
138 def make_pbsnodes_yaml(fin, fout):
139     """
140     read PBSNODES_ORIG_FILE sequentially and put in respective
          yaml file
141     """
142     global OfflineDownNodes
143
144     for line in fin:
```

```
145         line.strip()
146         searchdname = '^\w+([.-]?\w+)*'
147         if re.search(searchdname, line) is not None:   # line
                containing domain name
148             m = re.search(searchdname, line)
149             dname = m.group(0)
150             fout.write('domainname: ' + dname + '\n')
151 
152         elif 'state = ' in line:
153             nextchar = line.split()[2][0]
154             if nextchar == 'f':
155                 state = '-'
156             elif (nextchar == 'd') | (nextchar == 'o'):
157                 state = nextchar
158                 OfflineDownNodes += 1
159             else:
160                 state = nextchar
161             fout.write('state: ' + state + '\n')
162 
163         elif 'np = ' in line:
164             np = line.split()[2][0:]
165             # TotalCores = int(np)
166             fout.write('np: ' + np + '\n')
167 
168         elif 'jobs = ' in line:
169             ljobs = line.split('=')[1].split(',')
170             lastcore = 150000
171             for job in ljobs:
172                 core = job.strip().split('/')[0]
173                 if core == lastcore:
174                     print 'There are concurrent jobs assigned to
                            the same core!' + '\n' +' This kind of
                           Remapping is not implemented yet. Exiting
                           ..'
175                     sys.exit(1)
176                 job = job.strip().split('/')[1:][0].split('.')
                        [0]
177                 fout.write('- core: ' + core + '\n')
178                 fout.write('  job: ' + job + '\n')
179                 lastcore = core
180 
181         elif 'gpus = ' in line:
182             gpus = line.split(' = ')[1]
183             fout.write('gpus: ' + gpus + '\n')
184 
185         elif line.startswith('\n'):
186             fout.write('\n')
187 
188         elif 'ntype = PBS' in line:
```

```
189             print 'System currently not supported!'
190             sys.exit(1)
191
192
193     fin.close()
194     fout.close()
195
196
197 def read_pbsnodes_yaml(fin):
198     '''
199     extracts highest node number, online nodes
200     '''
201     global ExistingNodes, OfflineDownNodes, jobseries,
           BiggestWrittenNode, WNList, WNListRemapped, NodeNr,
           TotalCores, WorkingCores, AllWNsDict, AllWNsRemappedDict,
            HighestCoreBusy, MaxNP, NodeSubClusters, RemapNr,
           JUST_NAMES_FLAG
202
203     state = ''
204     for line in fin:
205         line.strip()
206         searchdname = 'domainname: ' + '(\w+-?\w+([.-]\w+)*)'
207         searchnodenr = '([A-Za-z0-9-]+)(?=\.|$)'
208         searchnodenrfind = '[A-Za-z]+|[0-9]+|[A-Za-z]+[0-9]+'
209         searchjustletters = '(^[A-Za-z-]+)'
210         if re.search(searchdname, line) is not None:   # line
               contains domain name
211             m = re.search(searchdname, line)
212             dname = m.group(1)
213             RemapNr += 1
214             '''
215             extract highest node number, online nodes
216             '''
217             ExistingNodes += 1    # nodes as recorded on
                   PBSNODES_ORIG_FILE
218             if re.search(searchnodenr, dname) is not None:  # if
                    a number and domain are found
219                 n = re.search(searchnodenr, dname)
220                 NodeInits = n.group(0)
221                 NameGroups = re.findall(searchnodenrfind,
                       NodeInits)
222                 NodeInits = '-'.join(NameGroups[0:-1])
223                 if NameGroups[-1].isdigit():
224                     NodeNr = int(NameGroups[-1])
225                 elif len(NameGroups) == 1: # if e.g. WN name is
                       just 'gridmon'
226                     if re.search(searchjustletters, dname) is
                           not None:  # for non-numbered WNs (eg.
                           fruit names)
```

```
227                         JUST_NAMES_FLAG += 1
228                         n = re.search(searchjustletters, dname)
229                         NodeInits = n.group(1)
230                         NodeNr += 1
231                         NodeSubClusters.add(NodeInits)    # for
                               non-uniform setups of WNs, eg g01...
                               and n01...
232                         AllWNsDict[NodeNr] = []
233                         AllWNsRemappedDict[RemapNr] = []
234                         if NodeNr > BiggestWrittenNode:
235                             BiggestWrittenNode = NodeNr
236                         WNList.append(NodeInits)
237                         WNList[:] = [UnNumberedWN.rjust(len(max(
                               WNList))) for UnNumberedWN in WNList
                               if type(UnNumberedWN) is str ]
238                         WNListRemapped.append(RemapNr)
239                     elif len(NameGroups) == 2 and not NameGroups
                        [-1].isdigit() and not NameGroups[-2].isdigit
                        ():
240                         NameGroups = '-'.join(NameGroups)
241                         if re.search(searchjustletters, dname) is
                               not None:  # for non-numbered WNs (eg.
                               fruit names)
242                             JUST_NAMES_FLAG += 1
243                             n = re.search(searchjustletters, dname)
244                             NodeInits = n.group(1)
245                             NodeNr += 1
246                             NodeSubClusters.add(NodeInits)    # for
                                   non-uniform setups of WNs, eg g01...
                                   and n01...
247                             AllWNsDict[NodeNr] = []
248                             AllWNsRemappedDict[RemapNr] = []
249                             if NodeNr > BiggestWrittenNode:
250                                 BiggestWrittenNode = NodeNr
251                             WNList.append(NodeInits)
252                             WNList[:] = [UnNumberedWN.rjust(len(max(
                                   WNList))) for UnNumberedWN in WNList
                                   if type(UnNumberedWN) is str ]
253                             WNListRemapped.append(RemapNr)
254                     elif NameGroups[-2].isdigit():
255                         NodeNr = int(NameGroups[-2])
256                     else:
257                         NodeNr = int(NameGroups[-3])
258                     # print 'NamedGroups are: ', NameGroups #####
                           DEBUGPRINT2
259                     NodeSubClusters.add(NodeInits)    # for non-
                           uniform setups of WNs, eg g01... and n01...
260                     AllWNsDict[NodeNr] = []
261                     AllWNsRemappedDict[RemapNr] = []
```

```
262                 if NodeNr > BiggestWrittenNode:
263                     BiggestWrittenNode = NodeNr
264                 if JUST_NAMES_FLAG <= 1:
265                     WNList.append(NodeNr)
266                 WNListRemapped.append(RemapNr)
267             elif re.search(searchjustletters, dname) is not None
                    :  # for non-numbered WNs (eg. fruit names)
268                 JUST_NAMES_FLAG += 1
269                 n = re.search(searchjustletters, dname)
270                 NodeInits = n.group(1)
271                 NodeNr += 1
272                 NodeSubClusters.add(NodeInits)    # for non-
                        uniform setups of WNs, eg g01... and n01...
273                 AllWNsDict[NodeNr] = []
274                 AllWNsRemappedDict[RemapNr] = []
275                 if NodeNr > BiggestWrittenNode:
276                     BiggestWrittenNode = NodeNr
277                 WNList.append(NodeInits)
278                 WNList[:] = [UnNumberedWN.rjust(len(max(WNList))
                        ) for UnNumberedWN in WNList]
279                 WNListRemapped.append(RemapNr)
280             else:
281                 NodeNr = 0
282                 NodeInits = dname
283                 AllWNsDict[NodeNr] = []
284                 AllWNsRemappedDict[RemapNr] = []
285                 NodeSubClusters.add(NodeInits)    # for non-
                        uniform setups of WNs, eg g01... and n01...
286                 if NodeNr > BiggestWrittenNode:
287                     BiggestWrittenNode = NodeNr + 1
288                 WNList.append(NodeNr)
289                 WNListRemapped.append(RemapNr)
290         elif 'state: ' in line:
291             nextchar = line.split()[1].strip("'")
292             if nextchar == 'f':
293                 state += '-'
294                 AllWNsDict[NodeNr].append('-')
295                 AllWNsRemappedDict[RemapNr].append('-')
296             else:
297                 state += nextchar
298                 AllWNsDict[NodeNr].append(nextchar)
299                 AllWNsRemappedDict[RemapNr].append(nextchar)
300
301         elif 'np:' in line:
302             np = line.split(': ')[1].strip()
303             AllWNsDict[NodeNr].append(np)
304             AllWNsRemappedDict[RemapNr].append(np)
305             if int(np) > int(MaxNP):
306                 MaxNP = int(np)
```

```
307             TotalCores += int(np)
308         elif 'core: ' in line:
309             core = line.split(': ')[1].strip()
310             WorkingCores += 1
311             if int(core) > int(HighestCoreBusy):
312                 HighestCoreBusy = int(core)
313         elif 'job: ' in line:
314             job = str(line.split(': ')[1]).strip()
315             AllWNsDict[NodeNr].append((core, job))
316             AllWNsRemappedDict[RemapNr].append((core, job))
317     HighestCoreBusy += 1
318 
319     '''
320     fill in non-existent WN nodes (absent from pbsnodes file)
            with '?' and count them
321     '''
322     if len(NodeSubClusters) > 1:
323         for i in range(1, RemapNr): # This RemapNr here is the
                LAST remapped node, it's the equivalent
                BiggestWrittenNode for the remapped case
324             if i not in AllWNsRemappedDict:
325                 AllWNsRemappedDict[i] = '?'
326     elif len(NodeSubClusters) == 1:
327         for i in range(1, BiggestWrittenNode):
328             if i not in AllWNsDict:
329                 AllWNsDict[i] = '?'
330 
331     if JUST_NAMES_FLAG <= 1:
332         WNList.sort()
333         WNListRemapped.sort()
334 
335     if min(WNList) > 9000 and type(min(WNList)) == int: # handle
            exotic cases of WN numbering starting VERY high
336         WNList = [element - min(WNList) for element in WNList]
337         options.BLINDREMAP = True
338     if len(WNList) < PERCENTAGE * BiggestWrittenNode:
339         options.BLINDREMAP = True
340 
341 
342 def make_qstatq_yaml(fin, fout):
343     global TotalRuns, TotalQueues  # qstatqLst
344     """
345     read QSTATQ_ORIG_FILE sequentially and put useful data in
            respective yaml file
346     """
347     Queuesearch = '^([a-zA-Z0-9_.-]+)\s+(--|[0-9]+[mgtkp]b[a-z
            ]*)\s+(--|\d+:\d+:?\d*)\s+(--|\d+:\d+:\d+)\s+(--)\s+(\d+)
            \s+(\d+)\s+(--|\d+)\s+([DE] R)'
348     RunQdSearch = '^\s*(\d+)\s+(\d+)'
```

```
349     for line in fin:
350         line.strip()
351         # searches for something like: biomed             --
                --    72:00:00   --   31   0 --   E R
352         if re.search(Queuesearch, line) is not None:
353             m = re.search(Queuesearch, line)
354             _, QueueName, Mem, CPUtime, Walltime, Node, Run,
                    Queued, Lm, State = m.group(0), m.group(1), m.
                    group(2), m.group(3), m.group(4), m.group(5), m.
                    group(6), m.group(7), m.group(8), m.group(9)
355             qstatqLst.append((QueueName, Run, Queued, Lm, State)
                    )
356             fout.write('- QueueName: ' + QueueName + '\n')
357             fout.write('  Running: ' + Run + '\n')
358             fout.write('  Queued: ' + Queued + '\n')
359             fout.write('  Lm: ' + Lm + '\n')
360             fout.write('  State: ' + State + '\n')
361             fout.write('\n')
362         elif re.search(RunQdSearch, line) is not None:
363             n = re.search(RunQdSearch, line)
364             TotalRuns, TotalQueues = n.group(1), n.group(2)
365     fout.write('---\n')
366     fout.write('Total Running: ' + str(TotalRuns) + '\n')
367     fout.write('Total Queued: ' + str(TotalQueues) + '\n')
368
369
370 def make_qstat_yaml(fin, fout):
371     """
372     read QSTAT_ORIG_FILE sequentially and put useful data in
            respective yaml file
373     """
374     firstline = fin.readline()
375     if 'prior' not in firstline:
376         UserQueueSearch = '^(([0-9-]+)\.([A-Za-z0-9-]+))\s+([A-
                Za-z0-9%_.=+/-]+)\s+([A-Za-z0-9.]+)\s+(\d+:\d+:?\d
                *|0)\s+([CWRQE])\s+(\w+)'
377         RunQdSearch = '^\s*(\d+)\s+(\d+)'
378         for line in fin:
379             line.strip()
380             # searches for something like: 422561.cream01
                                STDIN            see062
                    48:50:12 R see
381             if re.search(UserQueueSearch, line) is not None:
382                 m = re.search(UserQueueSearch, line)
383                 Jobid, Jobnr, CEname, Name, User, TimeUse, S,
                        Queue = m.group(1), m.group(2), m.group(3), m
                        .group(4), m.group(5), m.group(6), m.group(7)
                        , m.group(8)
384                 Jobid = Jobid.split('.')[0]
```

```
385                 fout.write('---\n')
386                 fout.write('JobId: ' + Jobid + '\n')
387                 fout.write('UnixAccount: ' + User + '\n')
388                 fout.write('S: ' + S + '\n')
389                 fout.write('Queue: ' + Queue + '\n')
390
391                 UserOfJobId[Jobid] = User # this actually
                        belongs to read_qstat() !
392                 fout.write('...\n')
393     elif 'prior' in firstline:
394         # e.g. job-ID  prior   name       user         state
                submit/start at     queue
                slots ja-task-ID
395         DIFFERENT_QSTAT_FORMAT_FLAG = 1
396         UserQueueSearch = '\s{2}(\d+)\s+([0-9]\.[0-9]+)\s+([A-Za
                -z0-9_.-]+)\s+([A-Za-z0-9._-]+)\s+([a-z])\s+(\d{2}/\d
                {2}/\d{2}|0)\s+(\d+:\d+:\d*|0)\s+([A-Za-z0-9_]+@[A-Za
                -z0-9_.-]+)\s+(\d+)\s+(\w*)'
397         RunQdSearch = '^\s*(\d+)\s+(\d+)'
398         for line in fin:
399             line.strip()
400             # searches for something like: 422561.cream01
                            STDIN            see062
                        48:50:12 R see
401             if re.search(UserQueueSearch, line) is not None:
402                 m = re.search(UserQueueSearch, line)
403                 Jobid, Prior, Name, User, State, Submit, StartAt
                        , Queue, QueueDomain, Slots, Ja_taskID = m.
                        group(1), m.group(2), m.group(3), m.group(4),
                         m.group(5), m.group(6), m.group(7), m.group
                        (8), m.group(9), m.group(10), m.group(11)
404                 print Jobid, Prior, Name, User, State, Submit,
                        StartAt, Queue, QueueDomain, Slots, Ja_taskID
405                 fout.write('---\n')
406                 fout.write('JobId: ' + Jobid + '\n')
407                 fout.write('UnixAccount: ' + User + '\n')
408                 fout.write('S: ' + State + '\n')
409                 fout.write('Queue: ' + Queue + '\n')
410
411                 UserOfJobId[Jobid] = User
412                 fout.write('...\n')
413
414
415 def read_qstat():
416     finr = open(QSTAT_YAML_FILE, 'r')
417     for line in finr:
418         if line.startswith('JobId:'):
419             JobIds.append(line.split()[1])
420         elif line.startswith('UnixAccount:'):
```

```
421             UnixAccounts.append(line.split()[1])
422         elif line.startswith('S:'):
423             Statuses.append(line.split()[1])
424         elif line.startswith('Queue:'):
425             Queues.append(line.split()[1])
426     finr.close()
427
428
429 def job_accounting_summary():
430     if len(NodeSubClusters) > 1 or options.BLINDREMAP:
431         print '=== WARNING: --- Remapping WN names and retrying
              heuristics... good luck with this... ---'
432     print '\nPBS report tool. Please try: watch -d ' + QTOPPATH
          + '. All bugs added by sfranky@gmail.com. Cross fingers
          now...\n'
433     print Colorize('===> ', '#') + Colorize('Job accounting
          summary', 'Nothing') + Colorize(' <=== ', '#') + Colorize
          ('(Rev: 3000 $) %s WORKDIR = to be added', '
          NoColourAccount') % (datetime.datetime.today()) #was:
          added\n
434     print 'Usage Totals:\t%s/%s\t Nodes | %s/%s  Cores |   %s+%s
           jobs (R + Q) reported by qstat -q' % (ExistingNodes -
          OfflineDownNodes, ExistingNodes, WorkingCores, TotalCores
          , int(TotalRuns), int(TotalQueues))
435     print 'Queues: | ',
436     if options.COLOR == 'ON':
437         for queue in qstatqLst:
438             if queue[0] in ColorOfAccount:
439                 print Colorize(queue[0], queue[0]) + ': ' +
                      Colorize(queue[1], queue[0]) + '+' + Colorize
                      (queue[2], queue[0]) + ' |',
440             else:
441                 print Colorize(queue[0], 'Nothing') + ': ' +
                      Colorize(queue[1], 'Nothing') + '+' +
                      Colorize(queue[2], 'Nothing') + ' |',
442     else:
443         for queue in qstatqLst:
444             print queue[0] + ': ' + queue[1] + '+' + queue[2] +
                  ' |',
445     print '* implies blocked\n'
446
447
448 def fill_cpucore_columns(value, CPUDict):
449     '''
450     Calculates the actual contents of the map by filling in a
          status string for each CPU line
451     '''
452     Busy = []
453
```

```
454     if value[0] == '?':
455         for CPULine in CPUDict:
456             CPUDict[CPULine] += '_'
457     else:
458         HAS_JOBS = 0
459         OwnNP = int(value[1])
460         OwnNPRange = [str(x) for x in range(OwnNP)]
461         OwnNPEmptyRange = OwnNPRange[:]
462 
463         for element in value[2:]:
464             if type(element) == tuple:  # everytime there is a
                    job:
465                 HAS_JOBS += 1
466                 Core, job = element[0], element[1]
467                 try:
468                     UserOfJobId[job]
469                 except KeyError, KeyErrorValue:
470                     print 'There seems to be a problem with the
                            qstat output. A JobID has gone rogue (
                            namely, ' + str(KeyErrorValue) +').
                            Please check with the System
                            Administrator.'
471                 CPUDict['Cpu' + str(Core) + 'line'] += str(
                        IdOfUnixAccount[UserOfJobId[job]])
472                 Busy.extend(Core)
473                 OwnNPEmptyRange.remove(Core)
474 
475         NonExistentCores = [item for item in MaxNPRange if item
                not in OwnNPRange]
476 
477         '''
478         the height of the matrix is determined by the highest-
                core WN existing. If other WNs have less cores,
479         these positions are filled with '#'s.
480         '''
481         for core in OwnNPEmptyRange:
482             CPUDict['Cpu' + str(core) + 'line'] += '_'
483         for core in NonExistentCores:
484                 CPUDict['Cpu' + str(core) + 'line'] += '#'
485 
486 
487 def insert(original, separator, pos, stopaftern = 0):
488     '''
489     insert separator into original (string) every posth position
            , optionally stopping after stopafter times.
490     '''
491     pos = int(pos)
492     if pos != 0: # default value is zero, means no vertical
            separators
```

```
493         sep = original[:]  # insert initial vertical separator
494         if stopaftern == 0:
495             times = len(original) / pos
496         else:
497             times = stopaftern
498         sep = sep[:pos] + separator + sep[pos:]
499         for i in range(2, times+1):
500             sep = sep[:pos * i + i-1] + separator + sep[pos * i
                + i-1:]
501         sep = separator + sep  # insert initial vertical
               separator
502         return sep
503     else: # no separators
504         return original
505
506
507 def calculate_Total_WNIDLine_Width(WNnumber): # (RemapNr) in
        case of multiple NodeSubClusters
508     '''
509     calculates the worker node ID number line widths (expressed
           by hxxxx's)
510     h1000 is the thousands' line
511     h0100 is the hundreds' line
512     and so on
513     '''
514     global h1000, h0100, h0010, h0001
515
516     if WNnumber < 10:
517         u_ = '123456789'
518         h0001 = u_[:WNnumber]
519
520     elif WNnumber < 100:
521         d_ = '0' * 9 + '1' * 10 + '2' * 10 + '3' * 10 + '4' * 10
                + '5' * 10 + '6' * 10 + '7' * 10 + '8' * 10 + '9' *
               10
522         u_ = '1234567890' * 10
523         h0010 = d_[:WNnumber]
524         h0001 = u_[:WNnumber]
525
526     elif WNnumber < 1000:
527         cent = int(str(WNnumber)[0])
528         dec = int(str(WNnumber)[1])
529         unit = int(str(WNnumber)[2])
530
531         c_ = str(0) * 99
532         for i in range(1, cent):
533             c_ += str(i) * 100
534         c_ += str(cent) * (int(dec)) * 10 + str(cent) * (int(
               unit) + 1)
```

```
535         h0100 = c_[:WNnumber]
536
537         d_ = '0' * 9 + '1' * 10 + '2' * 10 + '3' * 10 + '4' * 10
                + '5' * 10 + '6' * 10 + '7' * 10 + '8' * 10 + '9' *
               10
538         d__ = d_ + (cent - 1) * (str(0) + d_) + str(0)
539         d__ += d_[:int(str(dec) + str(unit))]
540         h0010 = d__[:WNnumber]
541
542         uc = '1234567890' * 100
543         h0001 = uc[:WNnumber]
544
545     elif WNnumber > 1000:
546         thou = int(str(WNnumber)[0])
547         cent = int(str(WNnumber)[1])
548         dec = int(str(WNnumber)[2])
549         unit = int(str(WNnumber)[3])
550
551         h1000 += str(0) * 999
552         for i in range(1, thou):
553             h1000 += str(i) * 1000
554         h1000 += str(thou) * ((int(cent)) * 100 + (int(dec)) *
               10 + (int(unit) + 1))
555
556         c_ = '0' * 99 + '1' * 100 + '2' * 100 + '3' * 100 + '4'
               * 100 + '5' * 100 + '6' * 100 + '7' * 100 + '8' * 100
                + '9' * 100
557         c__ = '0' * 100 + '1' * 100 + '2' * 100 + '3' * 100 + '4
               ' * 100 + '5' * 100 + '6' * 100 + '7' * 100 + '8' *
               100 + '9' * 100
558         h0100 = c_
559
560         for i in range(1, thou):
561             h0100 += c__
562         else:
563             h0100 += c__[:int(str(cent) + str(dec) +str(unit))
                   +1]
564
565         d_ = '0' * 10 + '1' * 10 + '2' * 10 + '3' * 10 + '4' *
               10 + '5' * 10 + '6' * 10 + '7' * 10 + '8' * 10 + '9'
               * 10
566         d__ = d_ * thou * 10  # cent * 10
567         d___ = d_ * (cent - 1)
568         h0010 = '0' * 9 + '1' * 10 + '2' * 10 + '3' * 10 + '4' *
                10 + '5' * 10 + '6' * 10 + '7' * 10 + '8' * 10 + '9'
                * 10
569         h0010 += d__
570         h0010 += d___
571         h0010 += d_[:int(str(dec) + str(unit)) + 1]
```

```
572
573         uc = '1234567890' * 1000
574         h0001 = uc[:WNnumber]
575
576
577 def find_Matrices_Width(WNnumber, WNList):
578     '''
579     masking/clipping functionality: if the earliest node number
            is high (e.g. 130), the first 129 WNs need not show up.
580     '''
581     Start = 0
582     if (options.MASKING is True) and min(WNList) >
            MIN_MASKING_THRESHOLD and type(min(WNList)) == str: # in
            case of named instead of numbered WNs
583         pass
584     elif (options.MASKING is True) and min(WNList) >
            MIN_MASKING_THRESHOLD and type(min(WNList)) == int:
585         Start = min(WNList) - 1   #exclude unneeded first empty
                nodes from the matrix
586     '''
587     Extra matrices may be needed if the WNs are more than the
            screen width can hold.
588     '''
589     if WNnumber > Start: # start will either be 1 or (masked >=
            MIN_MASKING_THRESHOLD + 1)
590         NrOfExtraMatrices = abs(WNnumber - Start + 10) /
                TermColumns
591     elif WNnumber < Start and len(NodeSubClusters) > 1: #
            Remapping
592         NrOfExtraMatrices = (WNnumber + 10) / TermColumns
593     else:
594         print "This is a case I didn't foresee (WNnumber vs
                Start vs NodeSubClusters)"
595
596     if UserCutMatrixWidth: # if the user defines a custom cut (
            in the configuration file)
597         Stop = Start + UserCutMatrixWidth
598         return (Start, Stop, WNnumber/UserCutMatrixWidth)
599     elif NrOfExtraMatrices: # if more matrices are needed due to
             lack of space, cut every matrix so that if fits to
            screen
600         Stop = Start + TermColumns - DEADWEIGHT
601         return (Start, Stop, NrOfExtraMatrices)
602     else: # just one matrix, small cluster!
603         Stop = Start + WNnumber
604         return (Start, Stop, 0)
605
606
```

```
607 def print_WN_ID_lines(start, stop, WNnumber): # WNnumber
    determines the number of WN ID lines needed  (1/2/3/4?)
608     global h1000, h0100, h0010, h0001
609     '''
610     h1000 is a header for the 'thousands',
611     h0100 is a header for the 'hundreds',
612     h0010 is a header for the 'tens',
613     h0001 is a header for the 'units' in the WN_ID lines
614     '''
615     global JUST_NAMES_FLAG
616     JustNameDict = {}
617     if JUST_NAMES_FLAG <= 1:  # normal case, numbered WNs
618         if WNnumber < 10:
619             print insert(h0001[start:stop], SEPARATOR, options.
                WN_COLON) + '={__WNID__}'
620 
621         elif WNnumber < 100:
622             print insert(h0010[start:stop], SEPARATOR, options.
                WN_COLON) + '={_Worker_}'
623             print insert(h0001[start:stop], SEPARATOR, options.
                WN_COLON) + '={__Node__}'
624 
625         elif WNnumber < 1000:
626             print insert(h0100[start:stop], SEPARATOR, options.
                WN_COLON) + '={_Worker_}'
627             print insert(h0010[start:stop], SEPARATOR, options.
                WN_COLON) + '={__Node__}'
628             print insert(h0001[start:stop], SEPARATOR, options.
                WN_COLON) + '={___ID___}'
629 
630         elif WNnumber > 1000:
631             print insert(h1000[start:stop], SEPARATOR, options.
                WN_COLON) + '={________}'
632             print insert(h0100[start:stop], SEPARATOR, options.
                WN_COLON) + '={_Worker_}'
633             print insert(h0010[start:stop], SEPARATOR, options.
                WN_COLON) + '={__Node__}'
634             print insert(h0001[start:stop], SEPARATOR, options.
                WN_COLON) + '={___ID___}'
635     elif JUST_NAMES_FLAG > 1 or options.FORCE_NAMES == True: #
        names (e.g. fruits) instead of numbered WNs
636         colour = 0
637         Highlight = {0: 'cmsplt', 1: 'Red'}
638         for line in range(len(max(WNList))):
639             JustNameDict[line] = ''
640         for column in range(len(WNList)): #was -1
641             for line in range(len(max(WNList))):
642                 JustNameDict[line] += Colorize(WNList[column][
                    line], Highlight[colour])
```

```
643             if colour == 1:
644                 colour = 0
645             else:
646                 colour = 1
647         for line in range(len(max(WNList))):
648             print JustNameDict[line] + '={__WNID__}'
649
650
651 def reset_yaml_files():
652     """
653     empties the files with every run of the python script
654     """
655     fin1temp = open(PBSNODES_YAML_FILE, 'w')
656     fin1temp.close()
657
658     fin2temp = open(QSTATQ_YAML_FILE, 'w')
659     fin2temp.close()
660
661     fin3temp = open(QSTAT_YAML_FILE, 'w')
662     fin3temp.close()
663
664 ################ MAIN ###################################
665
666 CONFIGFILE = os.path.expanduser(' /qtop/qtop/qtop.conf')
667 qtopconf = open(CONFIGFILE, 'r')
668 exec qtopconf
669
670 reset_yaml_files()
671 yamlstream1 = open(PBSNODES_YAML_FILE, 'a')
672 yamlstream2 = open(QSTATQ_YAML_FILE, 'a')
673 yamlstream3 = open(QSTAT_YAML_FILE, 'a')
674
675 if not os.path.getsize(PBSNODES_ORIG_FILE) > 0:
676     print 'Bailing out... Not yet ready for Sun Grid Engine
        clusters'
677     os.chdir(HOMEPATH + 'qt')
678     sys.exit(0)
679     # os.chdir('..')
680     # continue
681 else:
682     fin1 = open(PBSNODES_ORIG_FILE, 'r')
683 make_pbsnodes_yaml(fin1, yamlstream1)
684 yamlstream1 = open(PBSNODES_YAML_FILE, 'r')
685 read_pbsnodes_yaml(yamlstream1)
686 yamlstream1.close()
687
688 if not os.path.getsize(QSTATQ_ORIG_FILE) > 0:
689     print 'Your ' + QSTATQ_ORIG_FILE + ' file is empty! Please
        check your directory. Exiting ...'
```

```
690     os.chdir(HOMEPATH + 'qt')
691     sys.exit(0)
692     # os.chdir('..')
693     # continue
694 else:
695     fin2 = open(QSTATQ_ORIG_FILE, 'r')
696 make_qstatq_yaml(fin2, yamlstream2)
697 fin2.close()
698 yamlstream2.close()
699 
700 if not os.path.getsize(QSTAT_ORIG_FILE) > 0:
701     print 'Your ' + QSTAT_ORIG_FILE + ' file is empty! Please
            check your directory. Exiting ...'
702     os.chdir(HOMEPATH + 'qt')
703     sys.exit(0)
704     # os.chdir('..')
705     # continue
706 else:
707     fin3 = open(QSTAT_ORIG_FILE, 'r')
708 make_qstat_yaml(fin3, yamlstream3)
709 fin3.close()
710 yamlstream3.close()
711 # print dir
712 
713 read_qstat()
714 os.chdir(dir)
715 dir = os.getcwd()
716 
717 
718 #Calculation of split screen size
719 TermRows, TermColumns = os.popen('stty size', 'r').read().split
        ()
720 TermColumns = int(TermColumns)
721 
722 DEADWEIGHT = 15  # standard columns' width on the right of the
        CoreX map
723 
724 job_accounting_summary()
725 
726 # counting of R, Q, C attached to each user
727 RunningOfUser, QueuedOfUser, CancelledOfUser, WaitingOfUser,
        ExitingOfUser = {}, {}, {}, {}, {}
728 
729 for user, status in zip(UnixAccounts, Statuses):
730     if status == 'R':
731         RunningOfUser[user] = RunningOfUser.get(user, 0) + 1
732     elif status == 'Q':
733         QueuedOfUser[user] = QueuedOfUser.get(user, 0) + 1
734     elif status == 'C':
```

```
735         CancelledOfUser[user] = CancelledOfUser.get(user, 0) + 1
736     elif status == 'W':
737         WaitingOfUser[user] = WaitingOfUser.get(user, 0) + 1
738     elif status == 'E':
739         WaitingOfUser[user] = ExitingOfUser.get(user, 0) + 1
740
741 for account in RunningOfUser:
742     QueuedOfUser.setdefault(account, 0)
743     CancelledOfUser.setdefault(account, 0)
744     WaitingOfUser.setdefault(account, 0)
745     ExitingOfUser.setdefault(account, 0)
746
747 OccurenceDict = {}
748 for user in UnixAccounts:
749     OccurenceDict[user] = UnixAccounts.count(user)
750
751 Usersortedlst = sorted(OccurenceDict.items(), key=itemgetter(1),
        reverse=True)
752
753
754 '''
755 In case there are more users than the sum number of all numbers
        and
756 small/capital letters of the alphabet
757 '''
758 j = 0
759 if len(Usersortedlst)>62:
760     for i in xrange(62, len(Usersortedlst)+62):
761         POSSIBLE_IDS.append(str(i)[0])
762
763 for unixaccount in Usersortedlst:
764     IdOfUnixAccount[unixaccount[0]] = POSSIBLE_IDS[j]
765     j += 1
766
767 # this calculates and prints what is actually below the
768 # id|  R + Q /all | unix account etc line
769 for id in IdOfUnixAccount:
770     if id not in RunningOfUser:
771         RunningOfUser[id] = 0
772     if id not in QueuedOfUser:
773         QueuedOfUser[id] = 0
774     if id not in CancelledOfUser:
775         CancelledOfUser[id] = 0
776     if id not in WaitingOfUser:
777         WaitingOfUser[id] = 0
778     if id not in ExitingOfUser:
779         ExitingOfUser[id] = 0
780
781
```

```
782 for id in Usersortedlst:  # IdOfUnixAccount:
783     AccountsMappings.append([IdOfUnixAccount[id[0]],
            RunningOfUser[id[0]], QueuedOfUser[id[0]],
            CancelledOfUser[id[0]] + RunningOfUser[id[0]] +
            QueuedOfUser[id[0]] + WaitingOfUser[id[0]] +
            ExitingOfUser[id[0]], id])
784 AccountsMappings.sort(key=itemgetter(3), reverse=True)
785 ################################################
786
787
788 ### CPU lines ###################################
789 CPUCoreDict = {}
790 for i in range(MaxNP):
791     CPUCoreDict['Cpu' + str(i) + 'line'] = '' # Cpu0line,
            Cpu1line, Cpu2line, .. = '','','', ..
792     MaxNPRange.append(str(i))
793
794 if len(NodeSubClusters) > 1 or options.BLINDREMAP:
795     for _, WNProperties in zip(AllWNsRemappedDict.keys(),
            AllWNsRemappedDict.values()):
796         fill_cpucore_columns(WNProperties, CPUCoreDict)
797 elif len(NodeSubClusters) == 1:
798     for _, WNProperties in zip(AllWNsDict.keys(), AllWNsDict.
            values()):
799         fill_cpucore_columns(WNProperties, CPUCoreDict)
800
801 ### CPU lines ###################################
802
803
804 ############### Node State ######################
805 print Colorize('===> ', '#') + Colorize('Worker Nodes occupancy'
    , 'Nothing') + Colorize(' <=== ', '#') + Colorize('(you can
    read vertically the node IDs; nodes in free state are noted
    with - )', 'NoColourAccount')
806
807 '''
808 if there are non-uniform WNs in pbsnodes.yaml, e.g. wn01, wn02,
    gn01, gn02, ...,  remapping is performed
809 Otherwise, for uniform WNs, i.e. all using the same numbering
    scheme, wn01, wn02, ... proceed as normal
810 Number of Extra tables needed is calculated inside the
    calculate_Total_WNIDLine_Width function below
811 '''
812 if options.BLINDREMAP or len(NodeSubClusters) > 1:
813     calculate_Total_WNIDLine_Width(RemapNr)
814     for node in AllWNsRemappedDict:
815         NodeState += AllWNsRemappedDict[node][0]
816     (PrintStart, PrintEnd, NrOfExtraMatrices) =
            find_Matrices_Width(RemapNr, WNListRemapped)
```

```
817     print_WN_ID_lines(PrintStart, PrintEnd, RemapNr)
818 else: # len(NodeSubClusters) == 1 AND options.BLINDREMAP false
819     calculate_Total_WNIDLine_Width(BiggestWrittenNode)
820     for node in AllWNsDict:
821         NodeState += AllWNsDict[node][0]
822     (PrintStart, PrintEnd, NrOfExtraMatrices) =
           find_Matrices_Width(BiggestWrittenNode, WNList)
823     print_WN_ID_lines(PrintStart, PrintEnd, BiggestWrittenNode)
824
825
826 print insert(NodeState[PrintStart:PrintEnd], SEPARATOR, options.
       WN_COLON) + '=Node state'
827
828 ################ Node State ######################
829
830 for line in AccountsMappings:
831     if re.split('[0-9]+', line[4][0])[0] in ColorOfAccount:
832             AccountNrlessOfId[line[0]] = re.split('[0-9]+', line
                   [4][0])[0]
833     else:
834         AccountNrlessOfId[line[0]] = 'NoColourAccount'
835
836 AccountNrlessOfId['#'] = '#'
837 AccountNrlessOfId['_'] = '_'
838 AccountNrlessOfId[SEPARATOR] = 'NoColourAccount'
839
840 for ind, k in enumerate(CPUCoreDict):
841     ColourCPUCoreLst = list(insert(CPUCoreDict['Cpu' + str(ind)
           + 'line'][PrintStart:PrintEnd], SEPARATOR, options.
           WN_COLON))
842     ColourlessLineLen = len(''.join(ColourCPUCoreLst))
843     ColourCPUCoreLst = [Colorize(elem, AccountNrlessOfId[elem])
           for elem in ColourCPUCoreLst if elem in AccountNrlessOfId
           ]
844     line = ''.join(ColourCPUCoreLst)
845     #'''
846     #don't print the non-existent core lines in the first matrix
847     #(for when the remaining tables have machines with higher
           cores, but not the first matrix)
848     #'''
849     # if '\x1b[1;30m#\x1b[1;m' * ColourlessLineLen not in line:
850     print line + Colorize('=Core' + str(ind), 'NoColourAccount')
851
852
853 ############# Calculate remaining matrices #################
854 for i in range(NrOfExtraMatrices):
855     PrintStart = PrintEnd
856     if UserCutMatrixWidth:
857         PrintEnd += UserCutMatrixWidth
```

```
858     else:
859         PrintEnd += TermColumns - DEADWEIGHT #
860 
861     if options.BLINDREMAP or len(NodeSubClusters) > 1:
862         if PrintEnd >= RemapNr:
863             PrintEnd = RemapNr
864     else:
865         if PrintEnd >= BiggestWrittenNode:
866             PrintEnd = BiggestWrittenNode
867     print '\n'
868     if len(NodeSubClusters) == 1:
869         print_WN_ID_lines(PrintStart, PrintEnd,
                BiggestWrittenNode)
870     if len(NodeSubClusters) > 1:
871         print_WN_ID_lines(PrintStart, PrintEnd, RemapNr)
872     print insert(NodeState[PrintStart:PrintEnd], SEPARATOR,
            options.WN_COLON) + '=Node state'
873     for ind, k in enumerate(CPUCoreDict):
874         ColourCPUCoreLst = list(insert(CPUCoreDict['Cpu' + str(
                ind) + 'line'][PrintStart:PrintEnd], SEPARATOR,
                options.WN_COLON))
875         ColourlessLineLen = len(''.join(ColourCPUCoreLst))
876         ColourCPUCoreLst = [Colorize(elem, AccountNrlessOfId[
                elem]) for elem in ColourCPUCoreLst if elem in
                AccountNrlessOfId]
877         line = ''.join(ColourCPUCoreLst)
878         '''
879         if the first matrix has 10 machines with 64 cores, and
                the rest 190 machines have 8 cores, don't print the
                non-existent
880         56 cores from the next matrix on.
881         IMPORTANT: not working if vertical separators are
                present!
882         '''
883         if '\x1b[1;30m#\x1b[1;m' * ColourlessLineLen not in line
                :
884             print line + Colorize('=Core' + str(ind), '
                    NoColourAccount')
885 
886 
887 print Colorize('\n===> ', '#') + Colorize('User accounts and
    pool mappings', 'Nothing') + Colorize(' <=== ', '#') +
    Colorize('("all" includes those in C and W states, as
    reported by qstat)', 'NoColourAccount')
888 print ' id |  R   +   Q  /  all |    unix account | Grid
    certificate DN (this info is only available under elevated
    privileges)'
889 for line in AccountsMappings:
```

```
890     PrintString = '%3s | %4s + %4s / %4s | %15s |' % (line[0],
            line[1], line[2], line[3], line[4][0])
891     for account in ColorOfAccount:
892         if line[4][0].startswith(account) and options.COLOR == '
                ON':
893             PrintString = '%15s | %16s + %16s / %16s | %27s %4s'
                    % (Colorize(line[0], account), Colorize(str(line
                    [1]), account), Colorize(str(line[2]), account),
                    Colorize(str(line[3]), account), Colorize(line
                    [4][0], account), Colorize(SEPARATOR, '
                    NoColourAccount'))
894         elif line[4][0].startswith(account) and options.COLOR ==
                'OFF':
895             PrintString = '%2s | %3s + %3s / %3s | %14s |' %(
                    Colorize(line[0], account), Colorize(str(line[1])
                    , account), Colorize(str(line[2]), account),
                    Colorize(str(line[3]), account), Colorize(line
                    [4][0], account))
896         else:
897             pass
898     print PrintString
899
900 print '\nThanks for watching!'
901
902 os.chdir(dir)
```

qtop.py

qtop.colormap

```
1  ColorOfAccount = {
2      'Atlassm':   'Red_L',
3      'sgmatlas':  'Red_L',
4      'patlas':    'Red_L',
5      'satlas':    'Red_L',
6      'satl':  'Red_L',
7      'atlassgm':  'Red_L',
8      'atlsgm':    'Red_L',
9      'atlsg': 'Red_L',
10     'atlasusr':  'Red_L',
11     'atlass':    'Red_L',
12     'laspt': 'Red_L',
13     'atlpilot':  'Red_L',
14     'atpilot':   'Red_L',
15     'iatpilot':  'Red_L',
16     'atlasplt':  'Red_L',
17     'atlaspt':   'Red_L',
```

```
    'atlaspil':  'Red_L',
    'atlasplot': 'Red_L',
    'atlplot':   'Red_L',
    'atlpil':    'Red_L',
    'atlpl': 'Red_L',
    'atlaspilot': 'Red_L',
    'atlprod':   'Red',
    'atlasger':  'Red',
    'atlasde':   'Red',
    'datlas':    'Red',
    'atlasit':   'Red',
    'atlde': 'Red',
    'atlit': 'Red',
    'atlasprod': 'Red',
    'atlasprd':  'Red',
    'atlsprd':   'Red',
    'atlprd':    'Red',
    'atlpr': 'Red',
    'iatlas':    'Red',
    'iatprd':    'Red',
    'iatl':  'Red',
    'aatlpd':    'Red',
    'atlasfx':   'Red',
    'atlastw':   'Red',
    'atlx':  'Red',
    'atlp':  'Red',
    'atlu':  'Red',
    'atlhs': 'Red',
    'atlashs':   'Red',
    'atlasp':    'Red',
    'atlasfr':   'Red',
    'atlasana':  'Red',
    'Eatlas':     'Red',
    'shatlas':    'Red',
    'lcgatlas':   'Red',
    'atlasL':     'Red',
    'atlasil':    'Red',
    'atlanaly':   'Red',
    'atlas': 'Red',
    'atlbr': 'Red',
    'atls':  'Red',
    'atl':   'Red',
    'atfx':  'Red',
    'atprd': 'Red',
    'atcanpu':   'Red',
    'atcanpt':   'Red_L',
    'atcan': 'Red',
    'atcpu': 'Red',
    'atsgm': 'Red_L',
```

```
67      'pilatlas':  'Red_L',
68      'pilatl':    'Red_L',
69      'zipilatl':  'Red_L',
70      'patlit':    'Red_L',
71      'platl': 'Red_L',
72      'pltatlas':  'Red_L',
73      'pltatl':    'Red_L',
74      'atplt': 'Red_L',
75      'atlplt':    'Red_L',
76      'piatlas':   'Red_L',
77      'piatla':    'Red_L',
78      'piatl': 'Red_L',
79      'sgmatl':    'Red_L',
80      'zisgmatl':  'Red_L',
81      # sgmatl:   'Red',
82      'prdatlas':  'Red',
83      'prdatl':    'Red',
84      'prdat': 'Red',
85      'ziprdatl':  'Red',
86      'ziatlas':   'Red',
87      'patls': 'Red',
88      'patlit':    'Red',
89      'patl':  'Red',
90      'nordugrid-atlas': 'Red',
91      'nlv': 'Red_L',
92      'ncfk': 'Green_L',
93      'usatlas':   'Red',
94      # CMS VO commonly found pool account names
95      'cmssgm':    'Green_L',
96      'cmsplt':    'Green_L',
97      'pltcms':    'Green_L',
98      'cmspilot':  'Green_L',
99      'cmspil':    'Green_L',
100     'pilcms':    'Green_L',
101     'pcms':  'Green_L',
102     'sgmcms':    'Green_L',
103     'sgmcm': 'Green_L',
104     'scms':  'Green_L',
105     'priocms':   'Green_L',
106     'cmsprio':   'Green_L',
107     'pricms':    'Green_L',
108     'cmsprod':   'Green',
109     'cmsprd':    'Green',
110     'cmsmcp':    'Green',
111     'cmsnu': 'Green',
112     'cmst1prd': 'Green',
113     'cmprd': 'Green',
114     'uscmsPool': 'Green',
115     'uscms': 'Green',
```

```
116     'cmsp':  'Green',
117     'cmsusr':    'Green',
118     'cmsuwu':    'Green',
119     'cmsana':    'Green',
120     'cmsger':    'Green',
121     'cmss':  'Green',
122     'cmszu': 'Green',
123     'cmsau': 'Green',
124     'twcms': 'Green',
125     'cmsmu': 'Green',
126     'cms':   'Green',
127     'dcms':  'Green',
128     'prdcms':    'Green',
129     'icms':  'Green',
130     'cms': 'Green',
131     # ALICE VO commonly found pool account names
132     'alicesgm':  'Cyan',
133     'alice': 'Cyan',
134     'alisc': 'Cyan',
135     'alisgm':    'Cyan',
136     'alisg': 'Cyan',
137     'alibs': 'Cyan',
138     'alis':  'Cyan',
139     'alikn': 'Cyan',
140     'ali':   'Cyan',
141     'ialice':    'Cyan',
142     'salice':    'Cyan',
143     'sali':  'Cyan',
144     'caliceuser':    'Cyan',
145     'caliceusr': 'Cyan',
146     'calice':    'Cyan',
147     'calic': 'Cyan',
148     'sgmalice':  'Cyan',
149     'sgmali':    'Cyan',
150     # LHCb VO commonly found pool account names
151     'pdlhcb':    'Pink',
152     'prdlhcb':   'Pink',
153     'prdlhb':    'Pink',
154     'prdlhc':    'Pink',
155     'lhcbprd':   'Pink',
156     'lhbprd':    'Pink',
157     'lhprd': 'Pink',
158     'ilhcb': 'Pink',
159     'lhcbsgm':   'Purple',
160     'lhcbs': 'Purple',
161     'sgmlhcb':   'Purple',
162     'sgmlhb':    'Purple',
163     'sgmlhc':    'Purple',
164     'lhcbplt':   'Purple',
```

```
    'lhcbpilot': 'Purple',
    'lhpilot':   'Purple',
    'lhcpilot':  'Purple',
    'lhcbpil':   'Purple',
    'lhbpil':    'Purple',
    'pillhcb':   'Purple',
    'plhcb': 'Purple',
    'pilhcb':    'Purple',
    'pltlhcb':   'Purple',
    'pillhb':    'Purple',
    'pllhc': 'Purple',
    'tlhcb': 'Pink',
    'lhcbhs':    'Pink',
    'lhcbp': 'Pink',
    'lhcb':  'Pink',
    'lhcp':  'Pink',
    # dteam VO commonly found pool account names
    'dteamsgm':    'Brown',
    'dteamprd':    'Brown',
    'dteamuser': 'Brown',
    'dteamusr':  'Brown',
    'dteam': 'Brown',
    'dte':   'Brown',
    # OPS VO commonly found pool account names
    'opsplt':    'Yellow',
    'opsusr':    'Yellow',
    'opssgm':    'Yellow',
    'opssg': 'Yellow',
    'opsgm': 'Yellow',
    'sgmops':    'Yellow',
    'zisgmops':  'Yellow',
    'opsprd':    'Yellow',
    'opspil':    'Yellow',
    'pilops':    'Yellow',
    'samgrid':   'Yellow',
    'opss':  'Yellow',
    'sops':  'Yellow',
    'opsiber':   'Yellow',
    'opsib': 'Yellow',
    'ops':   'Yellow',
    #
    # Other VOs from the EGEE-I,II,III era
    'egee':  'Blue_L',
    # Biomed VO
    'biomedusr': 'Blue_L',
    'biomed':    'Blue_L',
    'biomd': 'Blue_L',
    'biome': 'Blue_L',
    'biocw': 'Blue_L',
```

```
214     'biostats':  'Blue_L',
215     'biotech':   'Blue_L',
216     'bio':   'Blue_L',
217     # Gear VO
218     'gearsgm':     'Blue_L',
219     'gearprd':    'Blue_L',
220     'gear':  'Blue_L',
221     # DECH VO
222     'dechsgm':     'Blue_L',
223     'dechprd':   'Blue_L',
224     'dechusr':   'Blue_L',
225     'dech':  'Blue_L',
226     # SEE VO
227     'seeops':    'Blue_L',
228     'seops': 'Blue_L',
229     'seegrid':   'Blue_L',
230     'seevo': 'Blue_L',
231     'see':   'Blue_L',
232     # ESR VO
233     'esrsgm':    'Blue_L',
234     'esr':   'Blue_L',
235     'earthscience':  'Blue_L',
236     # Fusion, auvergrid, compchem, enmr, voce, gaussian,
            balticgrid,
237     # Digital Media VOs
238     'fusionprd': 'Blue_L',
239     'fusionsgm': 'Blue_L',
240     'fusion':    'Blue_L',
241     'fusio': 'Blue_L',
242     'fusi':  'Blue_L',
243     'fusn':  'Blue_L',
244     'fus':   'Blue_L',
245     'auvergrid': 'Blue_L',
246     'enmr':  'Blue_L',
247     'complex':   'Blue_L',
248     'compchem':  'Blue_L',
249     'compc': 'Blue_L',
250     'compl': 'Blue_L',
251     'cmplx': 'Blue_L',
252     'vocesgm':   'Blue_L',
253     'voceprd':   'Blue_L',
254     'voce':  'Blue_L',
255     'gaussian':  'Blue_L',
256     'balticgrid': 'Blue_L',
257     'digmedia':  'Blue_L',
258     'dmedia':    'Blue_L',
259     # Hone VO
260     'prdhone':   'Cyan_L',
261     'prdhne':    'Cyan_L',
```

```
262     'prdhon':    'Cyan_L',
263     'honecker':  'Cyan_L',
264     'honeprd':   'Cyan_L',
265     'honesgm':   'Cyan_L',
266     'phone': 'Cyan_L',
267     'hone':  'Cyan_L',
268     'honp':  'Cyan_L',
269     # Other (High Energy) Physics VOs
270     'sixto': 'Cyan_L',
271     'sixt':  'Cyan_L',
272     'babaradm':  'Cyan_L',
273     'babarpro':  'Cyan_L',
274     'babar': 'Cyan_L',
275     'pheno': 'Cyan_L',
276     'dzerojim':  'Cyan_L',
277     'dzero': 'Cyan_L',
278     'dze':   'Cyan_L',
279     'dzerqa':    'Cyan_L',
280     'theophys_': 'Cyan_L',
281     'theophys':  'Cyan_L',
282     'zeususr':   'Cyan_L',
283     'zeus':  'Cyan_L',
284     'argo':  'Cyan_L',
285     'ilcprd':    'Cyan_L',
286     'ilcpr': 'Cyan_L',
287     'ilcp':  'Cyan_L',
288     'ilcusr':    'Cyan_L',
289     'ilcger':    'Cyan_L',
290     'ilc':   'Cyan_L',
291     'prdilc':    'Cyan_L',
292     'pilc':  'Cyan_L',
293     'sgmilc':    'Cyan_L',
294     'augersgm':  'Cyan_L',
295     'augerprd':  'Cyan_L',
296     'auger': 'Cyan_L',
297     'augp':  'Cyan_L',
298     'aug':   'Cyan_L',
299     'chatlas':    'Cyan_L',
300     'chcms':  'Cyan_L',
301     'chlhcb': 'Cyan_L',
302     'geant': 'Cyan_L',
303     'lhcft': 'Cyan_L',
304     'magic': 'Cyan_L',
305     'scier': 'Cyan_L',
306     'planck':    'Cyan_L',
307     'kaust': 'Brown',
308     # Extras; these are really randomly found user names; needed
            for screen
309     # clarity while in color more
```

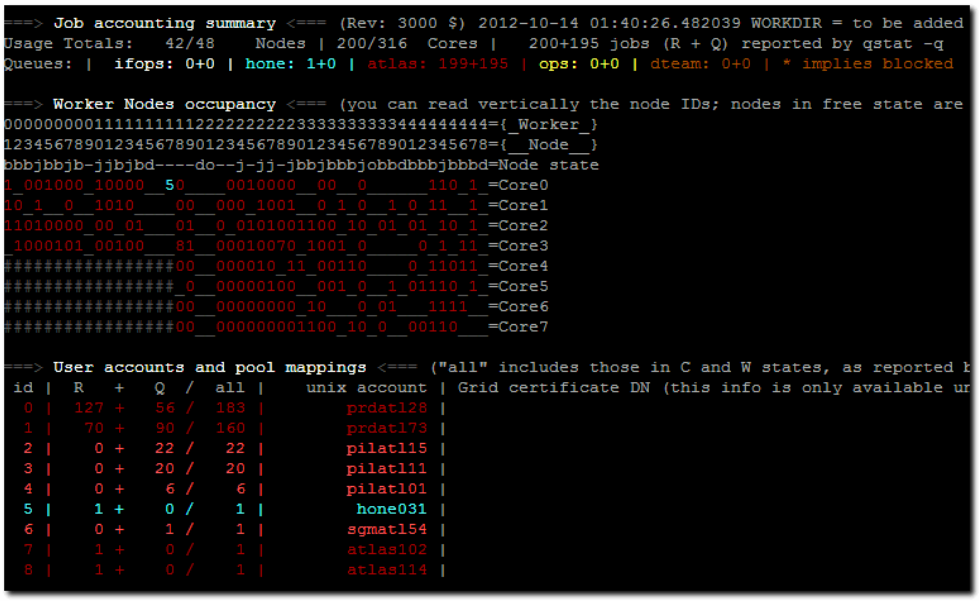
```
    'mwilli':      'Brown',
    'por':     'Blue',
    'dorisf':      'Cyan',
    'campoman':    'Brown',
    'gallet':      'Blue',
    'tlemmin':     'Cyan',
    'ls': 'Cyan',
    'train': 'Blue_L',
    'train': 'Blue_L',
    'pkoro':           'Brown',
    'fotis':           'Blue',
    'astrelchenko':    'Cyan',
    'tchristoudias':   'Brown',
    'dashed':    'Gray_L',
    'purple':    'Purple',
    'Red':    'Red',
    'DGray':   'Gray_D',
    'Brown':    'Brown',
    '_':    'Gray_D',
    '#':    'Gray_D',
    'NoColourAccount':    'normal',
    'Nothing':    'White'
    # catch-all rule for many more names
    #[[:alpha:]][-_[:alnum:].]* 'Gray_L',
}

CodeOfColor = {
    'Red_L': '1;31',
    'Red': '0;31',
    'Green_L': '1;32',
    'Green': '0;32',
    'Cyan': '0;36',
    'Pink': '1;35',
    'Purple': '0;35',
    'Brown': '0;33',
    'Yellow': '1;33',
    'Blue_L': '1;34',
    'Cyan_L': '1;36',
    'White': '1;37',
    'Blue': '0;34',
    'Gray_L': '1;37',
    'Gray_D':'1;30',
    'normal':   '0'
}
```

qtop.colormap

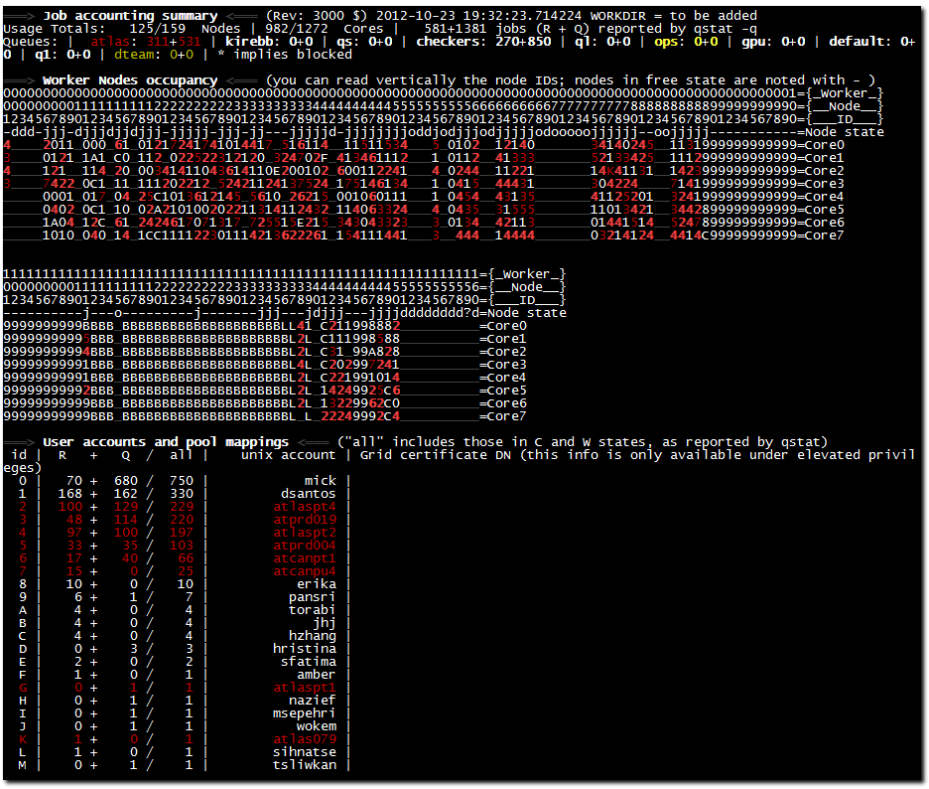
Παράρτημα Β΄

Παρουσίαση αποτελεσμάτων (Α)

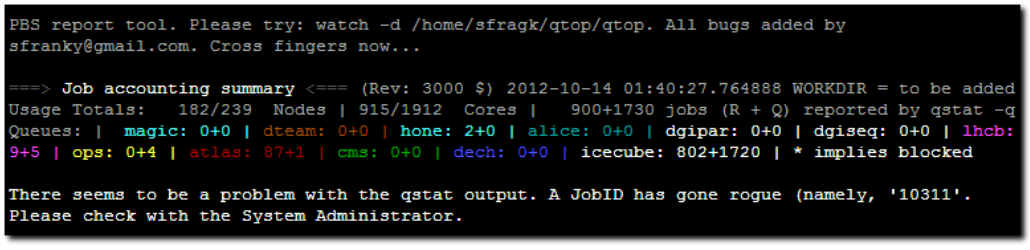
Εδώ παρουσιάζονται επιλεκτικά μερικές απεικονίσεις συστοιχιών στο ATLAS VO. Όπου είναι δυνατό, δίνεται και το resource URL. Οι εικόνες είναι πολύ μικρές για να είναι ευανάγνωστες, αλλά ο αναγνώστης μπορεί να μεγεθύνει (`Ctrl+"+"` για windows/linux ή `Cmd+"+"` στο mac) για να διακρίνει τα νούμερα.



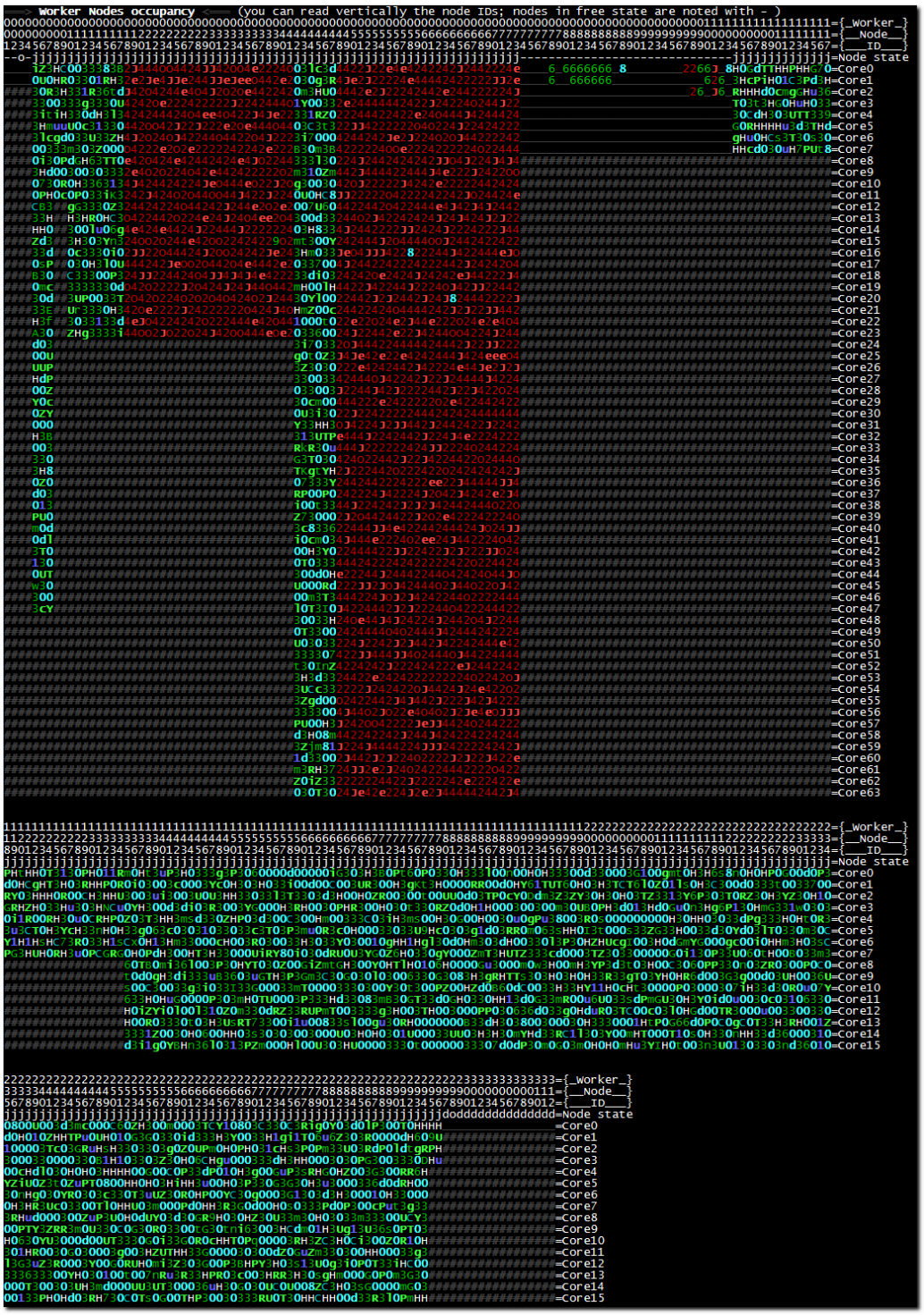
Σχήμα Β΄.1: *Αυτή είναι η πραγματική έξοδος του qtop, στην οποία βασίστηκε το παράδειγμα της ενότητας 2.4. Ξεχωρίζει η εργασία του χρήστη hone031 κωδικοποιημένη με το κυανό χρώμα και στους τρεις τομείς του qtop. Στον πρώτο τομέα φαίνεται άμεσα ότι ο χρήστης έχει μία εργασία να τρέχει και καμμία σε αναμονή, στο δεύτερο ότι τρέχει στον κόμβο 17 με αναγνωριστικό αριθμό 5 και στον τελευταίο η αντιστοίχιση χρώματος, αναγνωριστικού, λογαριασμού unix και τρεχούμενων/σε αναμονή/συνολικών εργασιών.*



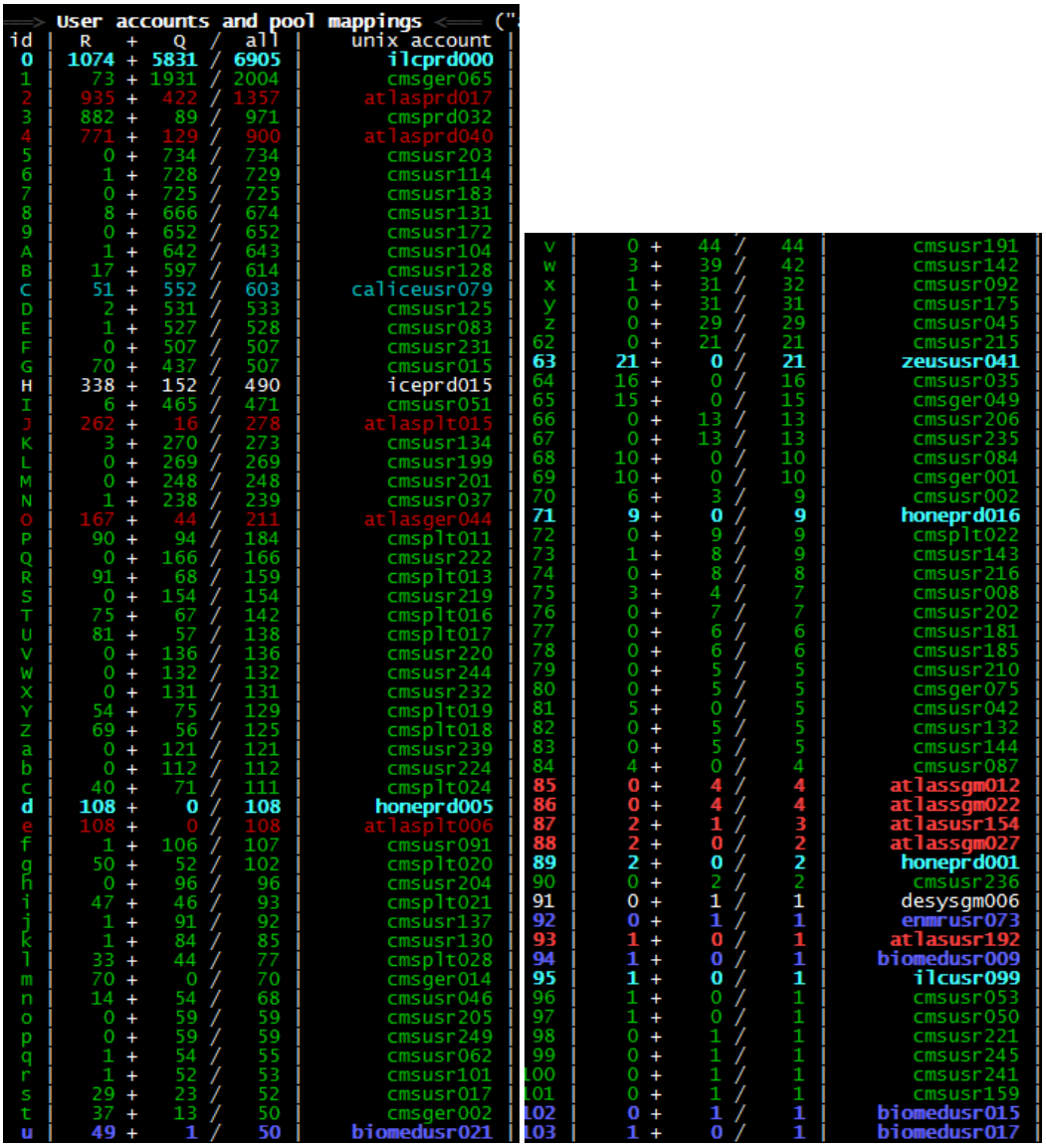
Σχήμα Β΄.2: *adm01.atlas.ualberta.ca:8443/cream-pbs-atlas*



Σχήμα Β΄.3: *Εδώ το qtop φανερώνει ένα bug του queueing συστήματος: λείπει το αναγνωριστικό μιας εργασίας από την έξοδο του* `qstat`*!*



Σχήμα Β΄.4: *grid-cr5.desy.de:8443/cream-pbs-atlas (Deutsches Elektronen Synchrotron, DESY). Στον πρώτο πίνακα ξεχωρίζουν οι κόμβοι που έχουν από 64 πυρήνες, ενώ υπάρχουν μερικοί που έχουν 8, 24, 48. Στους υπόλοιπους πίνακες φαίνονται μηχανήματα με 16 πυρήνες, με 8, ή και με 2. Το qtop κάνει εδώ «οικονομία χώρου», μικραίνοντας το ύψος των δύο τελευταίων πινάκων για να αντανακλάει τους λιγότερους πυρήνες.*



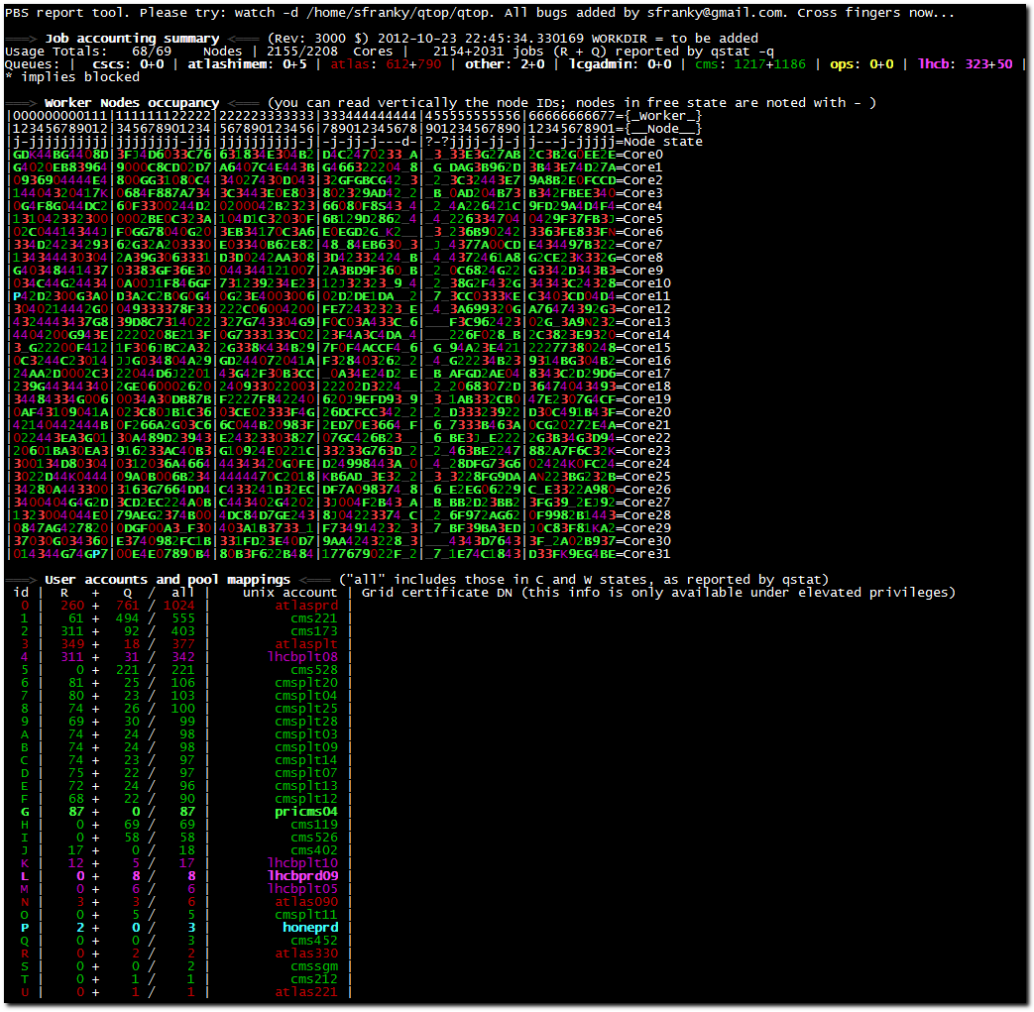
User accounts and pool mappings

id	R	+	Q	/	all	unix account
0	1074	+	5831	/	6905	ilcprd000
1	73	+	1931	/	2004	cmsger065
2	935	+	422	/	1357	atlasprd017
3	882	+	89	/	971	cmsprd032
4	771	+	129	/	900	atlasprd040
5	0	+	734	/	734	cmsusr203
6	1	+	728	/	729	cmsusr114
7	0	+	725	/	725	cmsusr183
8	8	+	666	/	674	cmsusr131
9	0	+	652	/	652	cmsusr172
A	1	+	642	/	643	cmsusr104
B	17	+	597	/	614	cmsusr128
C	51	+	552	/	603	caliceusr079
D	2	+	531	/	533	cmsusr125
E	1	+	527	/	528	cmsusr083
F	0	+	507	/	507	cmsusr231
G	70	+	437	/	507	cmsusr015
H	338	+	152	/	490	iceprd015
I	6	+	465	/	471	cmsusr051
J	262	+	16	/	278	atlasplt015
K	3	+	270	/	273	cmsusr134
L	0	+	269	/	269	cmsusr199
M	0	+	248	/	248	cmsusr201
N	1	+	238	/	239	cmsusr037
O	167	+	44	/	211	atlasger044
P	90	+	94	/	184	cmsplt011
Q	0	+	166	/	166	cmsusr222
R	91	+	68	/	159	cmsplt013
S	0	+	154	/	154	cmsusr219
T	75	+	67	/	142	cmsplt016
U	81	+	57	/	138	cmsplt017
V	0	+	136	/	136	cmsusr220
W	0	+	132	/	132	cmsusr244
X	0	+	131	/	131	cmsusr232
Y	54	+	75	/	129	cmsplt019
Z	69	+	56	/	125	cmsplt018
a	0	+	121	/	121	cmsusr239
b	0	+	112	/	112	cmsusr224
c	40	+	71	/	111	cmsplt024
d	108	+	0	/	108	honeprd005
e	108	+	0	/	108	atlasplt006
f	1	+	106	/	107	cmsusr091
g	50	+	52	/	102	cmsplt020
h	0	+	96	/	96	cmsusr204
i	47	+	46	/	93	cmsplt021
j	1	+	91	/	92	cmsusr137
k	1	+	84	/	85	cmsusr130
l	33	+	44	/	77	cmsplt028
m	70	+	0	/	70	cmsger014
n	14	+	54	/	68	cmsusr046
o	0	+	59	/	59	cmsusr205
p	0	+	59	/	59	cmsusr249
q	1	+	54	/	55	cmsusr062
r	1	+	52	/	53	cmsusr101
s	29	+	23	/	52	cmsusr017
t	37	+	13	/	50	cmsger002
u	49	+	1	/	50	biomedusr021
v	0	+	44	/	44	cmsusr191
w	3	+	39	/	42	cmsusr142
x	1	+	31	/	32	cmsusr092
y	0	+	31	/	31	cmsusr175
z	0	+	29	/	29	cmsusr045
62	0	+	21	/	21	cmsusr215
63	21	+	0	/	21	zeususr041
64	16	+	0	/	16	cmsusr035
65	15	+	0	/	15	cmsger049
66	0	+	13	/	13	cmsusr206
67	0	+	13	/	13	cmsusr235
68	10	+	0	/	10	cmsusr084
69	10	+	0	/	10	cmsger001
70	6	+	3	/	9	cmsusr002
71	9	+	0	/	9	honeprd016
72	0	+	9	/	9	cmsplt022
73	1	+	8	/	9	cmsusr143
74	0	+	8	/	8	cmsusr216
75	3	+	4	/	7	cmsusr008
76	0	+	7	/	7	cmsusr202
77	0	+	6	/	6	cmsusr181
78	0	+	6	/	6	cmsusr185
79	0	+	5	/	5	cmsusr210
80	0	+	5	/	5	cmsger075
81	5	+	0	/	5	cmsusr042
82	0	+	5	/	5	cmsusr132
83	0	+	5	/	5	cmsusr144
84	4	+	0	/	4	cmsusr087
85	0	+	4	/	4	atlassgm012
86	0	+	4	/	4	atlassgm022
87	2	+	1	/	3	atlasusr154
88	2	+	0	/	2	atlassgm027
89	2	+	0	/	2	honeprd001
90	0	+	2	/	2	cmsusr236
91	0	+	1	/	1	desysgm006
92	0	+	1	/	1	enmrusr073
93	1	+	0	/	1	atlasusr192
94	1	+	0	/	1	biomedusr009
95	1	+	0	/	1	ilcusr099
96	1	+	0	/	1	cmsusr053
97	1	+	0	/	1	cmsusr050
98	0	+	1	/	1	cmsusr221
99	0	+	1	/	1	cmsusr245
100	0	+	1	/	1	cmsusr241
101	0	+	1	/	1	cmsusr159
102	0	+	1	/	1	biomedusr015
103	1	+	0	/	1	biomedusr017

Σχήμα Β′.5: *Οι χρήστες της συστοιχίας του DESY. Εδώ, ο αριθμός χρηστών ξεπερνάει το μονοψήφιο επιτρεπτό όριο (26 πεζά + 26 κεφαλαία γράμματα της αγγλικής αλφαβήτου + 10 ψηφία), οπότε οι χρήστες που αντιστοιχούν σε διψήφιο αναγνωριστικό προς το παρόν αναπαρίστανται μόνο από τη «δεκάδα» του αναγνωριστικού τους.*

Σχήμα Β΄.6: *svr026.gla.scotgrid.ac.uk:8443/cream-pbs-atlas. Εδώ έχει εφαρμοστεί αυτόματα αναδιάταξη αρίθμησης (blind remapping), καθότι υπήρχαν αριθμοί που δε χρησιμοποιήθηκαν, και κόψιμο των ανύπαρκτων πυρήνων στο δεύτερο και τρίτο πίνακα.*

Σχήμα Β΄.7: *Εδώ φαίνεται πώς χειρίζεται το qtop τις συστοιχίες που δεν περιέχουν αριθμημένους κόμβους, αλλά ονομασμένους. Τα ονόματα μπαίνουν κατακόρυφα στη θέση αρίθμησης, και το χρώμα τους εναλλάσσεται ώστε να είναι ευανάγνωστα.*

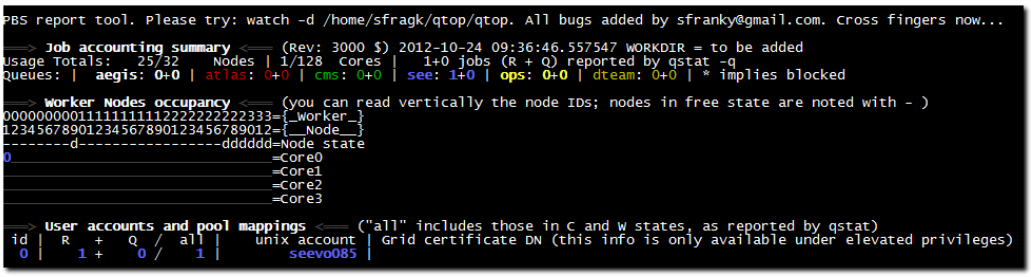


Σχήμα Β΄.8: *cream01.lcg.cscs.ch:8443/cream-pbs-atlashimem. Εδώ φαίνεται μία σχεδόν πλήρης συστοιχία που ανήκει στο ETH (Swiss Federal Institute of Technology). Έχουν προστεθεί κατακόρυφες γραμμές, χωρίζοντας τους κόμβους ανά 12άδες για αναγνωσιμότητα.*

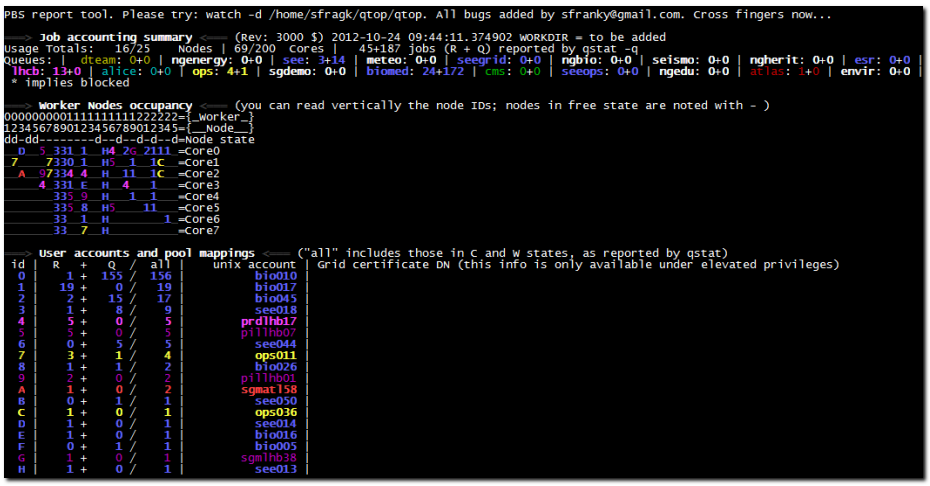
Παράρτημα Γ΄

Παρουσίαση αποτελεσμάτων (Β)

Εδώ παρουσιάζονται επιλεκτικά μερικές απεικονίσεις συστοιχιών στο SEE VO.

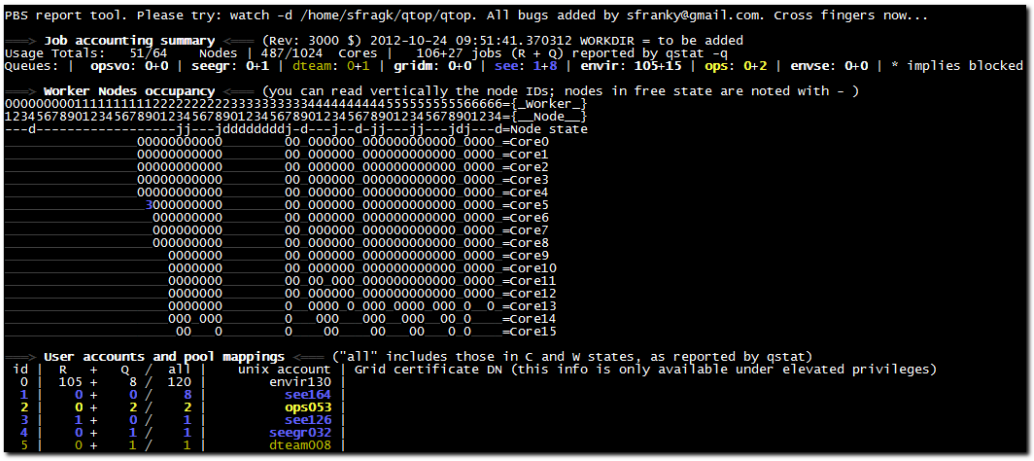


Σχήμα Γ΄.1: *ce-atlas.ipb.ac.png:2119/jobmanager-pbs-see*



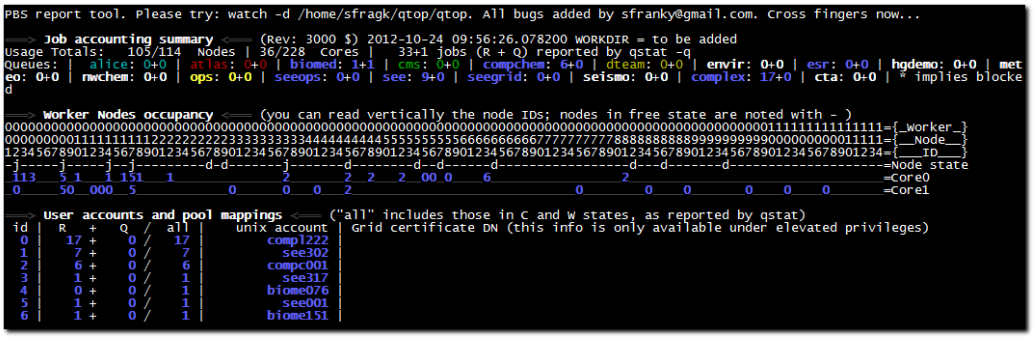
Σχήμα Γ΄.2: *ce_ngcc_acad_bg:2119/jobmanager-pbs-see*

Σχήμα Γ'.3: *ce01.marie.hellasgrid.gr:2119/jobmanager-pbs-see*



Σχήμα Γ'.4: *ce01.mosigrid.utcluj.ro:8443/cream-pbs-see*

Σχήμα Γ΄.5: *cox01.grid.metu.edu.tr:8443/cream-pbs-see*



Σχήμα Γ΄.6: *cream01.athena.hellasgrid.gr:8443/cream-pbs-see*

Βιβλιογραφία

[1] "A gentle Introduction to Grid Computing and Technologies", Rajkumar Buyya, Srikumar Venugopal, http://www.buyya.com/papers/GridIntro-CSI2005.pdf

[2] Ian Foster, Carl Kesselman, Steven Tuecke, The anatomy of the Grid, Enabling Scalable Virtual Organizations

[3] Αθανασία Χ. Ασσίκη, Ανάπτυξη συστοιχίας υπολογιστών τύπου GRID με χρήση ενδιάμεσου λογισμικού LCG-2_4_0 στο Εργαστήριο Φυσικής Υψηλών Ενεργειών, Διπλωματική Εργασία, Αθήνα Σεπτέμβριος 2005

[4] Επίσημη σελίδα Worldwide LHC Computing Grid http://wlcg.web.cern.ch

[5] Οδηγός του CERN για την τεχνολογία Πλέγματος, Grid Café, http://www.gridcafe.org/grid-architecture.html

[6] Survey and Analysis of Production Distributed Computing Infrastructures (arXiv:1208.2649v1 [cs.DC] 13 Aug 2012)

[7] Επίσημη σελίδα qtop http://fotis.web.cern.ch/fotis/QTOP

[8] pbsnodes, qstat, Linux man pages http://www.usc.edu/hpcc/pbsman/man8/pbsnodes.html, http://www.usc.edu/hpcc/pbsman/man1/qstat.html

[9] http://en.wikipedia.org/wiki/Grid_computing.html

[10] http://en.wikipedia.org/wiki/YAML

[11] "Think Python", http://www.greenteapress.com/thinkpython

[12] "Dive into Python", http://www.diveintopython.net

[13] "A Guide to LaTeX2e: document preparation for beginners and advanced users", Helmut Kopka and Patrick W. Daly, Addison-Wesley (1995).

[14] "Regular Expressions", http://en.wikipedia.org/wiki/Regular_expressions, http://www.regular-expressions.info