ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΜΑΚΕΔΟΝΙΑΣ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

Συστήματα Παράλληλης και
Κατανεμημένης Επεξεργασίας

PROCESSOR ALLOCATION IN PARALLEL SYSTEMS

http://arch.icte.uowm.gr

Μαμαλίγκα Μαριάννα 418
Τριανταφύλλου Άννα 456

Υπεύθυνος Καθηγητής: Δασυγένης Μηνάς

Η έννοια του Processor Allocation

- Η διαδικασία του Processor Allocation αναφέρεται στην κατανομή των επεξεργαστών ανάμεσα στις διάφορες διεργασίες που είναι διαθέσιμες στο σύστημα.
- Η παρούσα εργασία πραγματεύεται την κατανομή των επεξεργαστών στα παράλληλα συστήματα.

Στόχοι κατανομής επεξεργαστών

- Η κατανομή επεξεργαστών πρέπει να πληροί διάφορους εν μέρει αντιφατικούς στόχους:

1. **Υψηλό ποσοστό αξιοποίησης.** Η κατανομή επεξεργαστών πρέπει να μεγιστοποιεί την αξιοποίηση των πόρων, δηλαδή, θα πρέπει να αποφευχθεί κάθε είδους κατακερματισμός, έτσι ώστε όλοι οι επεξεργαστές να μπορούν να χρησιμοποιηθούν.
2. **Χαμηλή επιβάρυνση.** Δεδομένου ότι όλες οι αιτήσεις υποβάλλονται σε επεξεργασία κατά το χρόνο εκτέλεσης, οι αλγόριθμοι κατανομής των πόρων θα πρέπει να είναι γρήγοροι και να επιφέρουν μόνο χαμηλή επιβάρυνση.
3. **Επεκτασιμότητα.** Οι αλγόριθμοι πρέπει να υποστηρίζουν συστήματα χιλιάδων κόμβων χωρίς να δημιουργούν πρόβλημα (συμφόρηση).
4. **Χαμηλή καθυστέρηση.** Πρέπει να υποστηρίζονται χαμηλοί χρόνοι εκτέλεσης των παράλληλων προγραμμάτων. Σε ορισμένες μηχανές, ο χρόνος εκτέλεσης θα επηρεάζεται από το σύστημα ανάθεσης σε σχέση με το εύρος ζώνης επικοινωνίας και τις καθυστερήσεις εντός του partition.

Προγραμματιστικοί Αλγόριθμοι

- Η ανάθεση των επεξεργαστών πραγματοποιείται με βάση κάποιους προγραμματιστικούς αλγορίθμους (*scheduling algorithms*) [2].
- Οι προγραμματιστικοί αλγόριθμοι για τα συστήματα παράλληλης επεξεργασίας καθορίζουν (1) πόσες και ποιες εφαρμογές θα ενεργοποιηθούν και (2) πόσοι επεξεργαστές θα κατανεμηθούν στην κάθε μία. Οι αποφάσεις μπορεί να βασίζονται στις πληροφορίες που παρέχονται ή αποκτούνται για τις διαθέσιμες προς εκτέλεση εφαρμογές.
- Η χρήση της μνήμης, οι σύνδεσμοι επικοινωνίας και άλλοι πόροι μπορούν επίσης να επηρεάσουν την ενεργοποίηση και τις αποφάσεις κατανομής των επεξεργαστών.

Scheduling

- Το scheduling [7, 14, 12] πραγματοποιείται με βάση ένα κινούμενο παράθυρο, το μέγεθος του οποίου είναι ίσο με τον αριθμό των διεργασιών του συστήματος.
- Για παράδειγμα, αν μια εργασία δεν είναι εκτελέσιμη, διότι σχετίζεται με εργασίες οι οποίες βρίσκονται ακόμη στην ουρά, δεν θα προγραμματιστεί για εκτέλεση και ο διαθέσιμος επεξεργαστής θα ανατεθεί στην επόμενη εκτελέσιμη εργασία.
- Στο τέλος του χρόνου που αντιστοιχεί στην κάθε διεργασία, όλες οι εργασίες που έχουν ανατεθεί θα τοποθετηθούν στο τέλος της ουράς και οι επεξεργαστές θα προγραμματιστούν για εκτέλεση στο επόμενο παράθυρο με εκτελέσιμες εργασίες.

Δυναμικός και Στατικός Προγραμματισμός

Ανάλογα με το κόστος του processor rescheduling τρεις διαφορετικές προσεγγίσεις προγραμματισμού είναι δυνατές:

- **Δυναμικός Προγραμματισμός [1, 2]:** δεν υπάρχει κανένας περιορισμός στη συχνότητα του επαναπρογραμματισμού (rescheduling) των επεξεργαστών από μια εργασία σε άλλη. Ο scheduler μπορεί να πάρει έναν επεξεργαστή από την εκτελούμενη διεργασία και να τον αναθέσει σε μια άλλη διεργασία, όσο συχνά κρίνεται απαραίτητο.
- **Στατικός Προγραμματισμός [1,2]:** ένα σταθερό σύνολο επεξεργαστών είναι κατανεμημένο σε μια εργασία. Οι επεξεργαστές αυτοί απασχολούνται από μια εργασία καθ' όλη τη διάρκεια του χρόνου εκτέλεσής της και εγκαταλείπονται μόνο όταν η εργασία ολοκληρωθεί.

Δυναμικός και Στατικός Προγραμματισμός

- Το μειονέκτημα της στατικής πολιτικής ανάθεσης είναι ότι οδηγεί στο κατακερματισμό των πόρων των επεξεργαστών και στο συμβιβασμό της δικαιοσύνης. Επειδή η ανάθεση των εργασιών πραγματοποιείται κατά τη διάρκεια της εκτέλεσης, οι νέες αφίξεις υποχρεώνονται να περιμένουν στη σειρά μέχρι να γίνει διαθέσιμος κάποιος επεξεργαστής.
- Στις δυναμικές πολιτικές ανάθεσης από την άλλη μεριά, το σύστημα επιτρέπεται να απαντά σε αλλαγές του φόρτου, είτε αυτές προκαλούνται από νέες αφίξεις, είτε από ολοκλήρωση κάποιων εργασιών, είτε από αλλαγές στο παραλληλισμό των εκτελούμενων εργασιών. Και στις τρεις περιπτώσεις μια δυναμική πολιτική μπορεί να ανακατανέμει τους πόρους των επεξεργαστών, ώστε να ικανοποιήσει τις αλλαγές στο φόρτο του συστήματος.

Ημιστατικός Προγραμματισμός [1]

- Οι επεξεργαστές δεν είναι απαραίτητο να κατανέμονται σε εργασίες στατικά, αλλά η αλλαγή αυτών μεταξύ των εργασιών δεν είναι απεριόριστη.
- Ο scheduler πρέπει να έχει επίγνωση του κόστους που συνδέεται με τον επαναπρογραμματισμό των επεξεργαστών και να περιορίσει τις δραστηριότητες αυτές, έτσι ώστε η συνακόλουθη βελτίωση στην απόδοση να μην αντισταθμίζεται από το κόστος της εκτέλεσης των rescheduling εργασιών στο σύστημα.

Single-level και two-level scheduling

- Στο **single-level scheduling** [14], το λειτουργικό σύστημα είναι υπεύθυνο για τον προγραμματισμό όλων των εργασιακών νημάτων απευθείας στους επεξεργαστές. Το πρόβλημα είναι ότι απαιτεί υψηλούς πόρους του συστήματος και ίσως δεν έχει τη δυνατότητα ανταπόκρισης στις ανάγκες μιας εφαρμογής.
- Η λύση αυτού του προβλήματος βρίσκεται στη χρήση του **two-level scheduling** [14], όπου το λειτουργικό σύστημα είναι υπεύθυνο μόνο για την ανάθεση επεξεργαστών σε εργασίες, ενώ οι εφαρμογές πραγματοποιούν το προγραμματισμό των νημάτων οι ίδιες στους κατανεμημένους επεξεργαστές με τρόπο, ο οποίος ικανοποιεί τους κανόνες συγχρονισμού. Το εσωτερικό επίπεδο του scheduling επιτρέπει στην εφαρμογή να έχει περισσότερα νήματα από ότι οι κατανεμημένοι επεξεργαστές.

Διαμοιρασμός με βάση το χώρο και το χρόνο [8, 12]

- Όταν μια εργασία φτάνει σε ένα πολυεπεξεργαστικό σύστημα διαμοιραζόμενου χώρου, πρέπει να παρθεί μια απόφαση σχετικά με τον αριθμό και τις ιδιότητες των επεξεργαστών στους οποίους θα ανατεθεί.
- Μία προσαρμοστική πολιτική μπορεί να λάβει υπόψη της την κατάσταση του συστήματος κατά την εκκίνηση αλλά δεν επιτρέπει την παρέμβαση σε ενεργές διεργασίες.
- Μία δυναμική πολιτική διαχωρισμού μπορεί να χρησιμοποιήσει μία ή περισσότερες ενεργές διεργασίες για να εγκαθιδρύσει την νέα εκκίνηση.

Διαμοιρασμός με βάση το χώρο και το χρόνο

- Σε πολιτικές scheduling βασισμένες σε διαμοιρασμό χώρου, μια εισερχόμενη διεργασία αναθέτεται σε ένα υποσύνολο διαθέσιμων επεξεργαστών. Για αυτό, πολλές διεργασίες μπορεί να είναι ενεργές μέσα σε ένα πολυεπεξεργαστή ταυτόχρονα.
- Οι πολιτικές διαμοιρασμού χρόνου μπορεί να είναι είτε δυναμικές είτε προσαρμοστικές. Παράγοντες που επηρεάζουν την απόδοση των πολιτικών ανάθεσης επεξεργαστών σε πολυπρογραμματιστικά παράλληλα συστήματα είναι τα χαρακτηριστικά της επιτάχυνσης των εφαρμογών, η αρχιτεκτονική του πολυεπεξεργαστή και οι ιδιαίτερες απαιτήσεις συγκεκριμένου φόρτου εργασίας των εφαρμογών.

Προσεγγίσεις scheduling

- Με την προσαρμοστική ή **system state-based προσέγγιση προγραμματισμού** [1], ο αριθμός των εργασιών στο σύστημα χρησιμοποιείται ως μέτρο της κατάστασης του συστήματος: όσο μεγαλύτερος ο αριθμός των μόνιμων εργασιών, τόσο πιο απασχολημένο είναι το σύστημα. Ένα σταθερό σύνολο επεξεργαστών κατανέμεται σε μια εργασία που ξεκινά την εκτέλεση της.
- Σε κάθε δεδομένη χρονική στιγμή, ο αριθμός των επεξεργαστών ανά εργασία υπολογίζεται με βάση τον τρέχοντα αριθμό εργασιών στο σύστημα και για τους επεξεργαστές κατανεμημένους σε μια εργασία μπορεί να αλλάξει απαντώντας σε μια αλλαγή στον αριθμό των μόνιμων εργασιών.

Προσεγγίσεις scheduling

- Ενώ ο προσαρμοστικός προγραμματισμός προσπαθεί να βελτιώσει την απόδοση του συστήματος αντιδρώντας καταλλήλως σε μια αλλαγή κατάστασης του global system, ο **program behaviour-based προγραμματισμός** [1] υιοθετεί μια διαφορετική άποψη και παίρνει αποφάσεις επαναπρογραμματισμού βάσει μιας αλλαγής στη κατάσταση ή στη συμπεριφορά εκτέλεσης ενός προγράμματος.

Προσεγγίσεις scheduling

- Η διαδικασία **coscheduling** [2, 7] έχει προταθεί για να εξαλειφτεί το πρόβλημα του process thrashing από την ανάθεση των επεξεργαστών σε όλες τις συσχετιζόμενες εργασίες μιας εφαρμογής ταυτοχρόνως.
- Αν μια παράλληλη εφαρμογή έχει ένα σύνολο διεργασιών, οι οποίες αλληλεπιδρούν κατά την εκτέλεση, αυτές θα πρέπει να εκτελεστούν ταυτόχρονα.
- Όταν μια εφαρμογή είναι έτοιμη για εκτέλεση, οι διεργασίες της τοποθετούνται στην ουρά αναμονής του συστήματος.

Προσεγγίσεις scheduling

- Το **gang scheduling** [10, 11]είναι ένα σχήμα διαχείρισης πόρων για παράλληλα και κατανεμημένα συστήματα που συνδυάζουν διαμοιρασμό χρόνου και χώρου για να διασφαλίσουν υψηλή απόδοση συστήματος και μικρό χρόνο απόκρισης για ενεργές εργασίες.
- Ένας καλός scheduler εργασιών σε ένα πολυπρογραμματιστικό, παράλληλο περιβάλλον ισορροπεί την επιθυμία του χρήστη να ολοκληρώσει την εργασία όσο πιο γρήγορα γίνεται, ενώ ταυτόχρονα το σύστημα προσπαθεί να εξυπηρετήσει όσο το δυνατόν περισσότερες εργασίες.

Προσεγγίσεις scheduling

- Το gang scheduling αναπτύχθηκε αρχικά ώστε να υποστηρίζει τον fine-grain συγχρονισμό των παράλληλων εφαρμογών. Αλλά ένα μεγαλύτερο πλεονέκτημα είναι η προσφορά του στη μείωση της παρεμβολής.
- Το βασικό στοιχείο του gang scheduling είναι προσφέρει ένα παρόμοιο περιβάλλον με μια αφοσιωμένη μηχανή, στην οποία όλα τα νήματα μιας εργασίας προχωρούν μαζί, και ταυτόχρονα επιτρέπει τον διαμοιρασμό των πόρων.
- Το πρόβλημα του gang scheduling είναι ότι η απαίτηση, του ότι όλες οι εργασίες μίας διεργασίας θα πρέπει πάντα να τρέχουν μαζί, προκαλεί μεγάλο κατακερματισμό.

Προσεγγίσεις scheduling

- Το Backfilling [10] είναι μια βελτιστοποίηση, η οποία προσπαθεί να εξισορροπήσει τους στόχους της χρήσης της CPU και της διατήρησης της σειράς FCFS. Απαιτεί κάθε εργασία να διευκρινίζει το μέγιστο χρόνο εκτέλεσής της.
- Επίσης η τεχνική αυτή μπορεί να προκαλέσει καθυστέρηση στην εκτέλεση άλλων εργασιών που περιμένουν να εκτελεστούν (αυτές οι εργασίες δεν είναι πρώτες και για το λόγο αυτό δεν υπάρχουν κρατήσεις). Η προφανής εναλλακτική είναι να γίνονται κρατήσεις για όλες τις εργασίες. Αυτή η προσέγγιση ονομάζεται "conservative backfilling".
- Το backfilling εξαρτάται από τις εκτιμήσεις, για το πόσο χρόνο θα τρέχει μια εργασία, ώστε να αντιληφθεί πότε πρόσθετοι επεξεργαστές θα γίνουν διαθέσιμοι και για να διαπιστώσει πότε οι backfilled εργασίες θα τερματιστούν ώστε να μην παραβιαστεί κάποια κράτηση.

Αξιολόγηση των schedulers

- Η αξιολόγηση των παράλληλων schedulers [13] στηρίζεται σε δύο πράγματα: στη χρήση των απαραίτητων μετρικών και στη χρήση των απαραίτητων workloads πάνω στα οποία λειτουργεί ο scheduler.
- Εφόσον η απόδοση ενός υπολογιστικού συστήματος εξαρτάται από το workload, το οποίο επεξεργάζεται, μπορεί να χρησιμοποιηθεί ως benchmark ώστε να αξιολογήσει και να συγκρίνει τα πολλαπλά χαρακτηριστικά των schedulers στα παράλληλα supercomputers.
- Το Throughput είναι ένα καλό μέτρο απόδοσης για κλειστά συστήματα, διότι σε αυτά η άφιξη των διεργασιών εξαρτάται από την απόδοση του συστήματος. Κάθε διεργασία αναθέτεται ξανά αμέσως μόλις τερματιστεί. Έπειτα η ερώτηση τίθεται στο πόσο γρήγορα οι εργασίες επαναλαμβάνονται.

Αξιολόγηση των schedulers

- Η χρήση της CPU είναι ένα επίσης καλό μέτρο σε κάθε περίπτωση, διότι όσο καλύτερος είναι ο turnaround χρόνος των διεργασιών τόσο καλύτερη είναι και η απόδοση του συστήματος.
- Η ενέργεια είναι ένα ενδιαφέρον μετρικό καθώς ορίζεται από τη διαίρεση του throughput με το χρόνο απόκρισης, οπότε αυξάνεται εάν αυξάνεται το throughput, είτε όταν μειώνεται ο χρόνος απόκρισης. Επίσης εάν το throughput ρυθμίζεται από την εισερχόμενη διεργασία, η ενέργεια προσφέρει την ίδια πληροφορία με τον χρόνο απόκρισης.

Αξιολόγηση των schedulers

- Το makespan [13] είναι ένα μετρικό επιλογής για το off-line scheduling. Μπορεί να θεωρηθεί ως μια off-line εκδοχή του χρόνου απόκρισης. Είναι ο χρόνος στον οποίο ολόκληρος ο φόρτος εργασίας τερματίζεται και όχι ο χρόνος στον οποίο τερματίζεται η διεργασία. Σε ένα off-line σενάριο, συνδέεται άμεσα με τη χρήση της CPU και το throughput. Αυτό δεν ισχύει για το χρόνο απόκρισης ο οποίος εξαρτάται από την ουρά προγραμματισμού.
- Ο μέσος χρόνος απόκρισης είναι ένα ευρέως αποδεκτό μέτρο απόδοσης για ανοικτά, on-line συστήματα. Επίσης το μέτρο αυτό δίνει μεγαλύτερη έμφαση στις μακριές διεργασίες, σε σχέση με τις μικρές διεργασίες, οι οποίες εμφανίζονται συχνότερα.

Κατανεμημένο μοντέλο ανάθεσης [4]

(mesh-connected multicomputer)

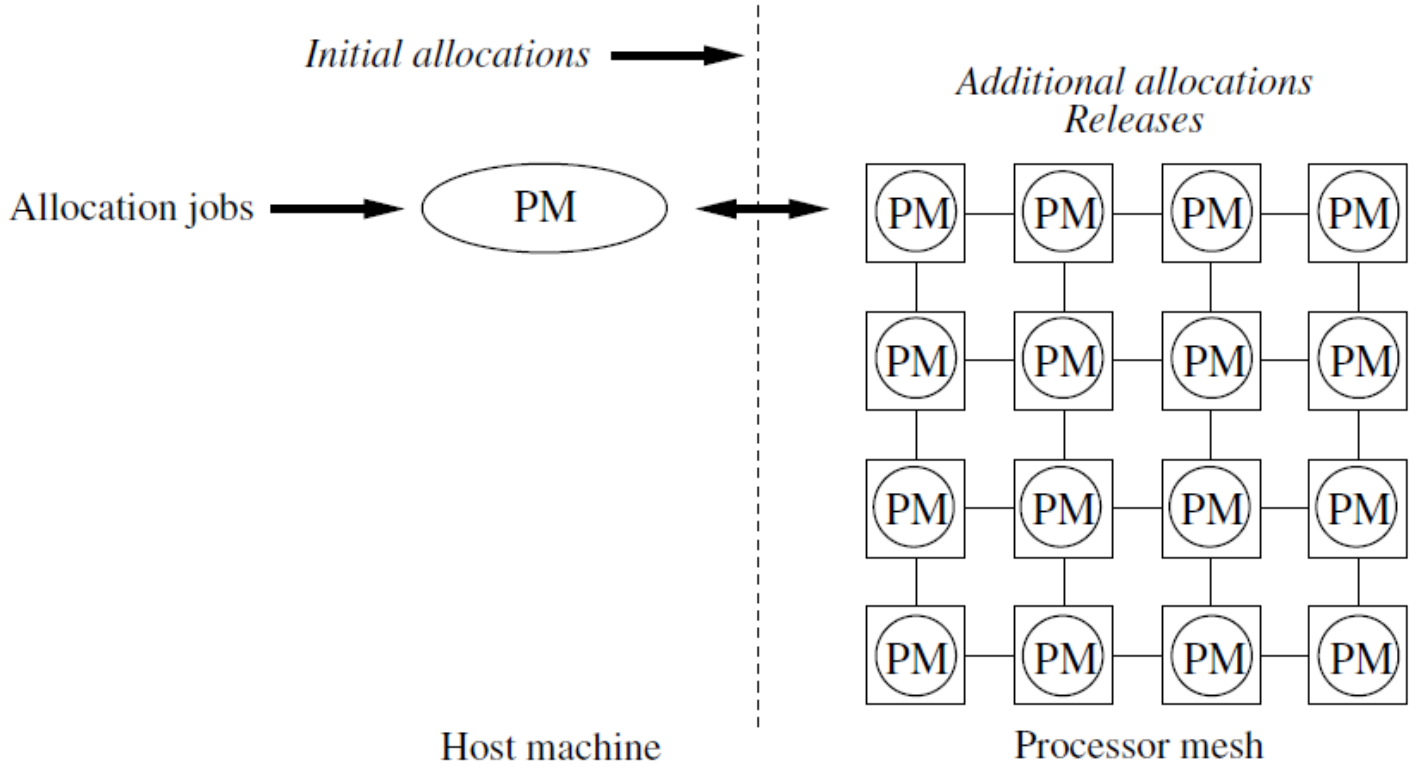

Fig. 1. Distributed allocation in a mesh-connected multicomputer.

Κατανεμημένο μοντέλο ανάθεσης
(mesh-connected multicomputer)

- Η εικόνα fig. 1 παρουσιάζει μια συνολική εικόνα του κατανεμημένου μοντέλου ανάθεσης και τους διανεμόμενους διαχειριστές επεξεργαστή (processor managers ή PMs) που συμμετέχουν στη λειτουργία της ανάθεσης.
- Κύριες διαφορές σε κεντρική διαχείριση είναι (i) η απουσία μιας κεντρικής δομής δεδομένων με πληροφορίες σχετικά με την κατάσταση όλων των επεξεργαστών, και (ii) η εκτέλεση των πράξεων ανάθεσης απευθείας στους κόμβους του μηχανήματος με ένα κατανεμημένο τρόπο, και όχι σε μια δομή δεδομένων στον host.
- Το μηχάνημα του host είναι πλέον υπεύθυνο μόνο για την ουρά εισερχόμενων αιτήσεων και τη διαβίβασή τους στο πλέγμα επεξεργαστών. Επικοινωνία μεταξύ του host και της μηχανής υφίσταται μέσω ενός κόμβου σύνορο. Ο κόμβος αυτός ονομάζεται ένα σημείο εισόδου (entry point), και λόγω του κατανεμημένου περιβάλλοντος αρκετά σημεία εισόδου μπορεί να χρησιμοποιηθούν για να βελτιώσουν την επεκτασιμότητα.

Κατανεμημένο μοντέλο ανάθεσης
(mesh-connected multicomputer)

- Κάθε κόμβος μηχανής έχει ένα τοπικό PM που είναι υπεύθυνος για την ανάθεση του επεξεργαστή. Οι PMs συνεργάζονται για να λύσουν το πρόβλημα ανάθεσης με ένα κατανεμημένο τρόπο. Η επικοινωνία μεταξύ των PMs και του εφαρμοσμένου κατανεμημένου σχήματος ανάθεσης εξαρτάται σε μεγάλο βαθμό από την αρχιτεκτονική του επιθυμητού μηχανήματος. Για να απλουστευθεί η αναζήτηση για δωρεάν πόρους, οι PMs σχηματίσουν μια λογική τοπολογία για να χαρτογραφηθεί αποτελεσματικά το δίκτυο διασύνδεσης του επιθυμητού μηχανήματος.

Noncontiguous Processor allocation

- Πολλοί από τους κατανεμημένους αλγόριθμους ανάθεσης εμφανίζουν υψηλό εξωτερικό κατακερματισμό (έως 30%), με αποτέλεσμα τη συμβιβαστική χρησιμοποίηση των μηχανημάτων. Ο κύριος λόγος για αυτή τη συμπεριφορά είναι ότι οι αλγόριθμοι αυτοί χρησιμοποιούν συνεχόμενη κατανομή (contiguous allocation) [4] για να μειώσουν τις προκύπτουσες παραμέτρους του partition, με σκοπό τη μείωση του κόστους της επικοινωνίας μέσα στο partition.
- Τρέχουσες τεχνολογίες επικοινωνίας, όπως η wormhole δρομολόγηση, επιτρέπουν να εξεταστούν μη συνεχόμενα συστήματα κατανομής (noncontiguous allocation schemes) [3, 4], δεδομένου ότι ο αριθμός των αλμάτων μεταξύ των κόμβων δεν είναι ο κυρίαρχος παράγοντας που καθορίζει την καθυστέρηση μηνυμάτων.
- Η χρήση μικρών partition των ελεύθερων επεξεργαστών σκορπισμένα σε όλο το μηχάνημα για το σχηματισμό μεγαλύτερων μη συνεχόμενων partition μειώνει σημαντικά τον εξωτερικό κατακερματισμό.

Noncontiguous Processor allocation

- Ωστόσο, σε μηχανές που συνδέονται σε δακτύλιο όπως τα SCI clusters, ή σε μηχανήματα με ιεραρχικά δίκτυα, η μη συνεχόμενη κατανομή εισάγει πιθανά προβλήματα διένεξης μηνυμάτων επειδή τα μηνύματα καταλαμβάνουν περισσότερες συνδέσεις, αποδίδοντας πιθανή παρεμβολή στην επικοινωνία με άλλες εργασίες.
- Εάν η γειτνίαση είναι ένα θέμα, προσπαθούμε να εξυπηρετήσουμε ένα αίτημα μαζί με τη συνεχόμενη κατανομή, και να ψάξουμε για μη συνεχόμενες προσθήκες μόνο κατά απαίτηση.
- Με αυτό τον τρόπο τα μη συνεχόμενα συστήματα θα πρέπει να θεωρηθούν ως μια προσθήκη, και όχι ως μια εναλλακτική λύση στην συνεχόμενη κατανομή.

Πολιτικές ανάθεσης επεξεργαστών

- Οι πολιτικές ανάθεσης πολυεπεξεργαστών κατηγοριοποιούνται σε time sharing and space sharing.
- Στο time sharing [16], οι εργασίες πολλαπλών διεργασιών ανατίθενται σε κάθε επεξεργαστή. Οι επεξεργαστές ταξινομούν τις διεργασίες τους κατά προτεραιότητα ή με βάση τον αλγόριθμο round robin.
- Στο space sharing [8, 16], ο πολυεπεξεργαστής χωρίζεται σε ανεξάρτητα υποσύνολα, όπου το καθένα περιέχει επεξεργαστικά στοιχεία που ονομάζονται partitions. Κάθε partition αναλαμβάνει και μια διεργασία.

Πολιτικές ανάθεσης επεξεργαστών

- Έχουν προταθεί πολλές λύσεις για το πρόβλημα του προγραμματισμού των πολυπρογραμματιζόμενων πολυεπεξεργαστών, όπως είναι το space sharing ή το space partitioning των επεξεργαστών, το gang scheduling των συσχετιζόμενων εργασιών, ο δυναμικός έλεγχος διεργασίας, ο μη παρεμποδιστικός συγχρονισμός και πολιτικές που εμποδίζουν εργασίες, οι οποίες διατηρούν ένα κλείδωμα, από το να διαγραφούν.
- Παρόλα αυτά, αυτές οι λύσεις απαιτούν να θυσιαστεί η γενική απόδοση του συστήματος για την βελτίωση της απόδοσης μιας μόνο εφαρμογής ή απαιτούν τροποποιήσεις στο υπάρχον λειτουργικό σύστημα συνδυασμένο με ένα ειδικό προγραμματιστικό μοντέλο.

Πολιτικές ανάθεσης επεξεργαστών

- Σε συστήματα που αποτελούνται από πολλά clusters επεξεργαστών τα οποία χρησιμοποιούν την τεχνική διαμοιρασμού χώρου για τον προγραμματισμό των διεργασιών, όπως είναι το Distributed ASCI1 Supercomputer (DAS) [9], μπορεί να χρειαστεί η τεχνική του **co-allocation** [5, 9], η οποία είναι η ταυτόχρονη ανάθεση των επεξεργαστών σε διεργασίες σε διαφορετικά clusters.

Πολιτικές ανάθεσης επεξεργαστών

- Για να είναι δυνατόν να παρθεί μια καλή απόφαση για το πότε θα υποβληθεί μια εφαρμογή σε ένα μόνο cluster ή σε πολλά clusters, θα ήταν βολικό να είχαμε μια σύνθετη εφαρμογή παραμετροποιημένη κατά τρόπο, που είναι χωρισμένη ανάμεσα στα clusters και κατά το σχέδιο επικοινωνίας της, ώστε να προσομοιώσει ένα εύρος από πιθανές εφαρμογές.

Πολιτικές ανάθεσης επεξεργαστών

- Η απόδοση του co allocation μετριέται μέσω προσομοιώσεων για το μέσο όρο απόδοσης των εργασιών που εξαρτώνται από ένα σύνολο προγραμματιστικών αποφάσεων όπως ο αριθμός των schedulers και των ουρών του συστήματος αλλά και μέσω του τρόπου με τον οποίο κατανέμονται εργασίες, που έχουν διαφορετικό αριθμό τμημάτων, ανάμεσα σε ουρές και προτεραιότητες που χρησιμοποιεί ο scheduler. Επίσης ένας ακόμη παράγοντας είναι η σύνθεση της ροής των εργασιών.

Πολιτικές ανάθεσης επεξεργαστών

- Χρησιμοποιώντας co-allocation δεν σημαίνει ότι όλες οι εργασίες πρέπει να χωριστούν σε components και να διαμοιραστούν στα διάφορα clusters. Μικρές εργασίες μπορούν επίσης να θεωρηθούν ως εργασίες που διαθέτουν ένα μόνο τμήμα και να τοποθετηθούν σε ένα μόνο cluster.

Πολιτικές ανάθεσης επεξεργαστών

- Σε ένα multicluster σύστημα όπου χρησιμοποιείται co-allocation, οι εργασίες μπορούν να είναι είτε single-component [5] είτε multi-component [5], και σε μια γενική περίπτωση και οι δύο τύποι είναι ταυτόχρονα παρόντες στο σύστημα.
- Είναι χρήσιμο να πραγματοποιείται ένας διαχωρισμός καθώς οι single-component εργασίες δεν χρησιμοποιούν co allocation ενώ οι multi-component εργασίες χρησιμοποιούν.

Πολιτικές ανάθεσης επεξεργαστών

- Σε ένα multicluster το οποίο χρησιμοποιεί co allocation και αντιμετωπίζει όλες τις αιτήσεις των εργασιών σαν εργασίες εκτός σειράς, ο χρήστης διευκρινίζει τον αριθμό των επεξεργαστών που χρειάζονται σε ξεχωριστά clusters αλλά όχι τα ίδια τα clusters. Επίσης βελτιώνει την απόδοση των single-component εργασιών με το να μην τις περιορίζει σε ένα μόνο cluster και διαλέγει ένα από όλα τα clusters του συστήματος στο οποίο χωράνε.
- Η καλύτερη επιλογή σύμφωνα με την αξιολόγηση διάφορων προγραμματιστικών αποφάσεων είναι να υπάρχει ένα σύστημα όπου θα αντιστοιχεί μόνο ένας scheduler για κάθε cluster και όλοι οι schedulers θα έχουν παγκόσμια πληροφόρηση ώστε να τοποθετούν τις εργασίες χρησιμοποιώντας co-allocation για ολόκληρο το σύστημα.

Πολιτικές ανάθεσης επεξεργαστών

- Το co-allocation με απαιτήσεις εκτός σειράς είναι μια καλή επιλογή για μεγάλες εργασίες, οι οποίες μπορούν να τρέξουν πιο γρήγορα εάν τις χωρίσουμε σε περισσότερα components και τις μοιράσουμε σε όλα τα clusters.
- Επίσης αντιμετωπίζει αποτελεσματικά και τις μικρές single-component εργασίες. Για ένα μεγάλο ποσοστό single-component εργασιών, που επιτρέπεται να τρέχουν πάνω σε οποιαδήποτε clusters, ακόμη και αν έχουν προγραμματιστεί από έναν μόνο παγκόσμιο scheduler, έχει αποδειχθεί να είναι μια καλύτερη επιλογή από το να τις διατηρούμε τοπικές σε ένα cluster.

Πολιτικές ανάθεσης επεξεργαστών

- Για multi-component εργασίες, έχοντας περισσότερους schedulers στο σύστημα και διαμοιράζοντας τις εργασίες ανάμεσά τους οδηγούμαστε στη βελτίωση της απόδοσης. Οποιαδήποτε από τις εργασίες στη αρχή των ουρών μπορεί να επιλεχθεί για να τρέξει εάν χωράει, πράγμα το οποίο ενεργοποιεί μια μορφή backfilling με παράθυρο ίσο με τον αριθμό των ουρών του συστήματος και αυξάνοντας την χρήση της CPU.

Πολιτικές ανάθεσης επεξεργαστών

- Η συνεχόμενη κατανομή έχει ερευνηθεί εκτενώς για τα 2D mesh multicomputers. Οι περισσότερες μελέτες έχουν επικεντρωθεί στη μείωση του υψηλού εξωτερικού κατακερματισμού που προκαλείται από τις συνεχόμενες στρατηγικές κατανομής. Παρακάτω περιγράφονται μερικές από αυτές τις στρατηγικές.
- Two-Dimensional Buddy System (2DBS) [3]: Η 2DBS κατανομή ισχύει για τα τετράγωνα συστήματα πλέγματος με τη δύναμη των δύο μηκών πλευρών. Οι επεξεργαστές που αναθέτονται σε εργασίες αποτελούν επίσης τετράγωνα υπο-πλέγματα με τη δύναμη των δύο μηκών πλευρών. Για παράδειγμα, αν μια εργασία ζητά 2 επεξεργαστές αναθέτεται ένα τετράγωνο τμήμα επεξεργαστών με μήκος πλευράς 2, με αποτέλεσμα 2 idle επεξεργαστές και έναν εσωτερικό κατακερματισμό του 50%. Η στρατηγική αυτή υποφέρει από υψηλό βαθμό κατακερματισμού. Επίσης, δεν μπορεί να χρησιμοποιηθεί για μη τετραγωνικά πλέγματα και δεν έχει πλήρη ικανότητα αναγνώρισης υπο-πλέγματος.

Πολιτικές ανάθεσης επεξεργαστών

- Frame Sliding (FS) [3]: Η στρατηγική ολισθαίνοντος πλαισίου ισχύει για κάθε πλέγμα, οποιουδήποτε μεγέθους και κάθε σχήμα υπο-πλέγματος. Ψάχνει για μια κατάλληλη κατανομή χρησιμοποιώντας ένα σύνολο ακολουθιακών μη επικαλυπτόμενων πλαισίων επεξεργαστών που καλύπτουν όλο το επιθυμητό πλέγμα.
- First Fit (FF) και Best Fit (BF) [3]: Τα ελεύθερα υπο-πλέγματα σαρώνονται και η First Fit στρατηγική αναθέτει το πρώτο υπο-πλέγμα που είναι αρκετά μεγάλο για να κρατήσει την εργασία, ενώ η Best Fit αναθέτει το μικρότερο κατάλληλο υπο-πλέγμα επιλέγοντας τη γωνία που έχει το μεγαλύτερο αριθμό απασχολημένων γειτόνων. Πίνακες bit χρησιμοποιούνται για τη σάρωση των διαθέσιμων επεξεργαστών.

Πολιτικές ανάθεσης επεξεργαστών

- Η μη συνεχόμενη κατανομή επιτρέπει οι εργασίες να εκτελούνται χωρίς να περιμένουν, όταν ο αριθμός των διαθέσιμων επεξεργαστών είναι επαρκής. Παρακάτω, περιγράφονται μερικές μη συνεχόμενες στρατηγικές κατανομής.
- Random [3]: Η τυχαία κατανομή είναι μια απλή στρατηγική στην οποία η αίτηση για ένα συγκεκριμένο αριθμό επεξεργαστών ικανοποιείται με τον ίδιο αριθμό επεξεργαστών που επιλέγονται τυχαία. Και οι δύο εσωτερικοί και εξωτερικοί κατακερματισμοί εξαλείφονται εφόσον όλες οι εργασίες λάβουν ακριβώς τον απαιτούμενο αριθμό των επεξεργαστών που έχει ζητηθεί.
- Paging [3]: Στη στρατηγική σελιδοποίησης, ολόκληρο το 2D πλέγμα χωρίζεται σε σελίδες που είναι υπο-πλέγματα με μήκος ίσων πλευρών 2size_index, όπου size_index είναι ένας θετικός ακέραιος. Μια σελίδα είναι η μονάδα κατανομής.
- Multiple Buddy System (MBS) [3]: Το MBS είναι μια επέκταση της στρατηγικής 2D buddy. Κατά την εκκίνηση, το δίκτυο πλέγματος διαιρείται σε μη επικαλυπτόμενα τετράγωνα υπό-πλέγματα με μήκη πλευράς που είναι δυνάμεις του 2.

Πολιτικές ανάθεσης επεξεργαστών

- **ASP (Adaptive Static Partitioning)** [15]: όταν μια εργασία φτάσει, διαμοιράζεται στους idle επεξεργαστές του συστήματος σύμφωνα με τον παραλληλισμό της. Αν κανένας επεξεργαστής δεν είναι διαθέσιμος, η εργασία μπαίνει στη σειρά πίσω από τις εργασίες που ήδη περιμένουν. Όταν μια εργασία ολοκληρωθεί, οι επεξεργαστές που ελευθερώνονται κατανέμονται στις εργασίες που περιμένουν, μέσω του αλγορίθμου round robin και με τον περιορισμό ότι καμία εργασία δεν παίρνει περισσότερους επεξεργαστές από αυτούς που επιτρέπει το επίπεδο παραλληλισμού της.
- Για παράδειγμα, αν υπάρχουν πέντε εργασίες που περιμένουν και μια που μόλις έχει ολοκληρωθεί απασχολώντας 100 επεξεργαστές, τότε ο επιτρεπόμενος παραλληλισμός ανά εργασία είναι (50,25,100,10,10), οπότε η ανάθεση θα πραγματοποιηθεί ως εξής (28, 25,27,10,10).
- Η συγκεκριμένη πολιτική έχει δείξει να έχει υψηλότερη απόδοση από ότι αρκετές στατικές πολιτικές ανάθεσης, κάτω από συγκεκριμένες παραμέτρους.

Πολιτικές ανάθεσης επεξεργαστών

- **FCFS (First Come First Served)** [2, 15]: κάθε εργασία διαμοιράζεται σε επεξεργαστές, όποτε κάποιος γίνεται διαθέσιμος, μέχρι εκεί που επιτρέπει το επίπεδο παραλληλισμού της. Οι επεξεργαστές που ελευθερώνονται, από μια εργασία που ολοκληρώθηκε, αναθέτονται πρώτα σε μία εργασία που βρίσκεται σε εκτέλεση και όπου η ανάθεσή της είναι μικρότερη από το διαθέσιμο παραλληλισμό της και ύστερα σε εργασίες που περιμένουν στη σειρά για να εκτελεστούν.
- Με τη συγκεκριμένη μέθοδο το παραπάνω παράδειγμα μας δίνει ως αποτέλεσμα ανάθεσης το εξής (50,25,25,0,0).

Πολιτικές ανάθεσης επεξεργαστών

- **EQ (Dynamic Equiallocation**) [15]: οι πολιτικές αυτές αναθέτουν ένα ίσο μερίδιο επεξεργαστικής δύναμης σε κάθε εργασία του συστήματος, εκτός εάν μια εργασία έχει μικρότερο διαθέσιμο παραλληλισμό από την τιμή της ίσης ανάθεσης. Στη συγκεκριμένη περίπτωση κάθε εργασία αναθέτεται σε όσους επεξεργαστές επιτρέπει ο παραλληλισμός και η τιμή τής ίσης ανάθεσης υπολογίζεται ξανά για τις υπόλοιπες εργασίες.
- Η ανάθεση των εργασιών του προηγούμενου παραδείγματος με βάση τη πολιτική αυτή είναι η εξής (27.5,25,27.5,10,10). Η ανάθεση της ενέργειας που ανακατανέμεται συμβαίνει όταν φτάνει μια νέα εργασία, όταν μια εργασία ολοκληρώνεται και όταν συμβαίνουν αλλαγές στο διαθέσιμο παραλληλισμό μιας εργασίας.
- Κάτω από τις ίδιες απαιτήσεις, χωρίς καμία συσχέτιση μεταξύ παραλληλισμού και απαιτήσεων αλλά και μέσω γραμμικών speedups, οι πολιτικές FCFS, EQ και PSAPF αποδίδουν περίπου το ίδιο.

Πολιτικές ανάθεσης επεξεργαστών

- **EQS (Spatial EQuiallocation**) [15]: είναι μια EQ πολιτική κατά την οποία η επεξεργαστική δύναμη κατανέμεται χωρικά, πρώτα ολικά και έπειτα τμηματικά.
- Για παράδειγμα, εάν μια εργασία είναι να λάβει μια ανάθεση 27.5 μονάδων επεξεργαστικής δύναμης, τότε αναθέτεται σε 27 επεξεργαστές και λαμβάνει επιπλέον 0.5 μονάδες επεξεργαστικής ισχύος από το διαμοιρασμό χρόνου ενός πρόσθετου επεξεργαστή.

Πολιτικές ανάθεσης επεξεργαστών

- **PSAPF (Preemptive Smallest Available Parallelism First**) [15]: η κεντρική σειρά εργασιών είναι μια παρεμποδιστική σειρά, η οποία είναι οργανωμένη σε αύξουσα σειρά του διαθέσιμου παραλληλισμού των εργασιών. Οι εργασίες με τον ίδιο διαθέσιμο παραλληλισμό εκτελούνται με βάση τη διαδικασία first-come-first-served. Όπως συμβαίνει στην πολιτική FCFS, κάθε διεργασία κατανέμεται σε επεξεργαστές που γίνονται διαθέσιμοι μέχρι το μέγιστο επίπεδο που επιτρέπει ο διαθέσιμος παραλληλισμός της.
- Επίσης οι επεξεργαστές που ελευθερώνονται από την ολοκλήρωση μιας εργασίας αρχικά τοποθετούνται σε μια εργασία που ήδη εκτελείται και όπου η ανάθεσή της είναι μικρότερη από το διαθέσιμο παραλληλισμό της και ύστερα σε εργασίες που περιμένουν στη σειρά για να εκτελεστούν.

Πολιτικές ανάθεσης επεξεργαστών

- Η ανάθεση των εργασιών του προηγούμενου παραδείγματος με βάση τη πολιτική αυτή είναι η εξής (50,25,5,10,10). Η ανάθεση της ενέργειας που ανακατανέμεται συμβαίνει όταν φτάνει μια νέα εργασία, όταν μια εργασία ολοκληρώνεται και όταν συμβαίνουν αλλαγές στο διαθέσιμο παραλληλισμό μιας εργασίας.
- Κάτω από υπερβολικά εξαίρετες συνθήκες αλλά και συγκεκριμένους παράλληλους διαχωρισμούς, η πολιτική PSAPF έχει παρατηρηθεί πως παρουσιάζει υψηλή απόδοση όταν υπάρχει μεγάλη συσχέτιση μεταξύ απαιτήσεων και παραλληλισμού.

Πολιτικές ανάθεσης επεξεργαστών

- Οι πολιτικές αυτές ορίζονται από την επεξεργαστική δύναμη που αναθέτουν σε εργασίες που βρίσκονται στη σειρά και όχι από την ανάθεση των επεξεργαστών σε ξεχωριστές δοκιμασίες μέσα σε μία εργασία. Επίσης όλες οι παραπάνω κάνουν χρήση του διαθέσιμου παραλληλισμού των εργασιών για να αποφασίσουν πόσους επεξεργαστές θα αναθέσουν σε κάθε εργασία του συστήματος.
- Βέβαια, μέσα από διάφορες μελέτες έχει παρατηρηθεί πως δεν υπάρχουν κάποιες συγκεκριμένες παράμετροι που ρυθμίζουν την απόδοση της κάθε πολιτικής ή το πώς κάθε πολιτική συγκρίνει τη γενική ζήτηση των εργασιών ή το διαθέσιμο παραλληλισμό ή τις καμπύλες του speedup που κυμαίνονται από μερικώς γραμμικές σε γραμμικές.

Πολιτικές ανάθεσης επεξεργαστών

- Η πολιτική ASP, σύμφωνα με προηγούμενες μελέτες, έχει παρατηρηθεί να έχει υψηλή απόδοση σε σχέση με άλλες στατικές πολιτικές ανάθεσης.
- Η πολιτική FCFS είναι μια πολύ απλή πολιτική και παρουσιάζει υψηλή απόδοση για συγκεκριμένα workloads.
- Η πολιτική EQ έχει επίσης υψηλή απόδοση για συγκεκριμένα workloads.
- Η πολιτική EQS είναι μια ιδανική, χωρική equialllocation πολιτική.

Πολιτικές ανάθεσης επεξεργαστών

- Η συσχέτιση της διακύμανσης στη ζήτηση μπορεί να είναι χρήσιμη στις αποφάσεις που αφορούν την απόδοση μιας πολιτικής. Αυτό μπορεί να είναι προφανές από τα προγραμματιστικά αποτελέσματα ενός μονού επεξεργαστή, αλλά προηγούμενες αναλύσεις και συγκρίσεις των πολιτικών των παράλληλων επεξεργαστών έχουν διαπιστώσει την ανάγκη ιδιαίτερων απαιτήσεων ή ιδιαίτερων εργασιών που αφορούν το χρόνο.

Πολιτικές ανάθεσης επεξεργαστών

- Ο ρυθμός εκτέλεσης και η συσχέτιση ανάμεσα σε απαιτήσεις και παραλληλισμό είναι επίσης παράμετροι που επηρεάζουν την απόδοση μιας πολιτικής. Καθώς οι καμπύλες του speedup έχουν προσδιοριστεί σε προηγούμενες μελέτες, σε πολλές περιπτώσεις δεν χρειάζονται καν να ληφθούν μέτρα που έχουν να κάνουν με τη συσχέτιση. Αυτό το αποτέλεσμα δείχνει ότι είναι σημαντικό να λαμβάνεται υπόψη η συσχέτιση ανάμεσα σε απαιτήσεις και παραλληλισμό, όπως επίσης και οι υπαινιγμοί για την απόδοση μιας πολιτικής.
- Για ένα συγκεκριμένο σύνολο εργασιών με μια συνάρτηση, η οποία έχει έναν μη φθίνοντα ρυθμό εκτέλεσης, η πολιτική EQS επιτυγχάνει μια βέλτιστη χρήση της CPU σε σχέση με όλες τις υπόλοιπες πολιτικές.

Πολιτικές ανάθεσης επεξεργαστών

- Η πολιτική EQS ουσιαστικά υπερτερεί της πολιτικής ASP για ασυσχέτιστα και συσχετιζόμενα workloads.
- Η πολιτική EQS επίσης υπερτερεί της πολιτικής ASP, όταν η συσχέτιση της διακύμανσης στη ζήτηση είναι μέτρια προς υψηλή ακόμη και όταν η συσχέτιση του φόρτου εργασίας είναι υψηλή.
- Η πολιτική PSAPF έχει μικρότερο μέσο όρο απόκρισης χρόνου από την πολιτική EQ μόνο για υψηλά συσχετιζόμενους φόρτους εργασίας σε χαμηλές με μέτριες τιμές της συσχέτισης της διακύμανσης στη ζήτηση, όταν η εκτέλεση πλησιάζει στο να γίνει γραμμική. Τα αποτελέσματα αυτά ισχύουν για όλες τις διανομές του διαθέσιμου παραλληλισμού και γενικά τις διανομές των απαιτήσεων.

Πολιτικές ανάθεσης επεξεργαστών

- Η πολιτική PSAPF υπερτερεί της πολιτικής FCFS για τις περισσότερες παραμέτρους χώρου.
- Από τη στιγμή που τα υπολογιστικά συστήματα γενικής χρήσης είναι πιθανό να έχουν υψηλή διαφοροποίηση στις απαιτήσεις που έχουν να κάνουν με την επεξεργασία εργασιών, η πολιτική EQS μοιάζει να είναι η καλύτερη επιλογή για εκτέλεση ανάμεσα στις πολιτικές που προαναφέρθηκαν.

Πολιτικές ανάθεσης επεξεργαστών

- **Loop-Level Process Control (LLPC)** [7]: μια απλή και αποτελεσματική στρατηγική ανάθεσης εργασιών για πολυπρογραμματιστικούς πολυεπεξεργαστές, η οποία αποτελεί μια επέκταση προηγούμενων πολιτικών ελέγχου διεργασιών που είναι συμβατές με τα υπάρχοντα λειτουργικά συστήματα και το προγραμματιστικό μοντέλο fork-join.

- μοντέλο fork-join [7]: στην αρχή ενός παράλληλου τμήματος του προγράμματος οι υπο-διεργασίες θα διαγραφούν για να δημιουργηθούν διεργασίες παιδιά μέσω της εντολής fork του λειτουργικού συστήματος Unix.

Πολιτικές ανάθεσης επεξεργαστών

- Στην αρχή κάθε παράλληλου τμήματος το οποίο ανήκει σε μια εφαρμογή, το LLPC χρησιμοποιεί τον παρόντα φόρτο του συστήματος για να αποφασίσει το άνω όριο του αριθμού των διεργασιών το οποίο μπορεί να παράγει η εφαρμογή για αυτό το παράλληλο τμήμα.
- Βασιζόμενοι στο Perfect Club Fortran benchmarks [7] μπορούμε να πούμε ότι το LLPC έχει τη δυνατότητα να παράγει υψηλή χρήση συστήματος διατηρώντας επίσης υψηλή απόδοση για κάθε εφαρμογή.

Πολιτικές ανάθεσης επεξεργαστών

- Ακόμη ένα πλεονέκτημα αυτής της στρατηγικής είναι ότι δεν είναι ορατή στο προγραμματιστή και δεν απαιτεί τροποποιήσεις από το λειτουργικό σύστημα.
- Συνεπώς, η εφαρμογή μπορεί να παραμείνει προσβάσιμη και συμβατή.
- Παρά το γεγονός ότι αυτή η προσέγγιση δεν μπορεί να εκμεταλλευτεί το μέγιστο του παραλληλισμού σε μια εφαρμογή, μπορεί να αυξήσει την συμβατότητα του κώδικα και να ελαφρύνει το βάρος του προγραμματιστή στην δημιουργία ενός νέου παράλληλου αλγορίθμου.

Πολιτικές ανάθεσης επεξεργαστών

- Το loop-level process control βοηθάει στην βελτίωση της επίδοσης των εφαρμογών σε time-shared και multiprogrammed multiprocessor περιβάλλοντα.
- Όταν ο φόρτος του συστήματος είναι υψηλός , το LLPC προσπαθεί να μειώσει το context switching rate με το να επιτρέπει στις εφαρμογές να δημιουργούν μόνο ένα μικρό αριθμό από διεργασίες αντί για τον μέγιστο αριθμό διεργασιών που πιθανώς θα θέλουν να χρησιμοποιήσουν.

Πολιτικές ανάθεσης επεξεργαστών

- Σαν αποτέλεσμα, η εκτέλεση μπορεί να προχωρήσει για όλες τις εφαρμογές καθώς πλέον δεν υπάρχει μια μόνο εφαρμογή η οποία να μονοπωλεί όλους τους επεξεργαστές.
- Το πιο σημαντικό θέμα είναι πώς να αποφασιστεί ο φόρτος με ελάχιστο overhead του παρόντος συστήματος. Ορίζουμε τον φόρτο του συστήματος ως τον αριθμό των απασχολημένων επεξεργαστών συν το συνολικό αριθμό των διεργασιών που περιμένουν στην ουρά εργασίας.
- Αν το σύστημα είναι απασχολημένο, το LLPC επιτρέπει σε μια εφαρμογή να δημιουργήσει τουλάχιστον μια διεργασία.

Πολιτικές ανάθεσης επεξεργαστών

- Το LLPC δεν αναστέλλει καμία ενεργή διεργασία ακόμη και αν ο φόρτος του συστήματος είναι υψηλός.
- Παρά το ότι το LLPC δεν μπορεί να προσαρμοστεί άμεσα στις αλλαγές του φόρτου του συστήματος, οι διεργασίες διακόπτονται αυτόματα στο τέλος κάθε παράλληλου βρόγχου και η ανάθεση κάθε επεξεργαστή επανεξετάζεται σε κάθε είσοδο βρόγχου.
- Σαν αποτέλεσμα, η στρατηγική αυτή θα προσαρμοστεί στο φόρτο του συστήματος μέσα σε ένα μικρό χρονικό διάστημα.
- Επίσης μη επιτρέποντας τη δυναμική διακοπή διεργασιών απλοποιείται η εκτέλεση της πολιτικής αυτής και δεν απαιτούνται αλλαγές στο scheduler του συστήματος, το οποίο είναι ένα σημαντικό πλεονέκτημα για τα υπάρχοντα λειτουργικά συστήματα.

Πολιτικές ανάθεσης επεξεργαστών [8]

Table 1. Comparison of the strategies (job point-of-view).

	Information used	Average time savings per job (minutes)	Fraction of jobs that rebel	Average time savings per rebel (minutes)	Fraction of rebels that lose
STUB	A	8.8	37%	23.8	34%
HEUR	$A,R(n)$	12.8	37%	34.6	68%
PRED	$A,R(n),E[Q(n)]$	12.8	8%	160	8%
BIAS	$A,R(n),E[Q(n)],\beta_0,\beta_1$	13.5	10%	135	8%
OPT	$A,R(n),Q(n)$	13.8	14%	98.6	0%

Πολιτικές ανάθεσης επεξεργαστών

- STUB [8], είναι η στρατηγική κατά την οποία οι χρήστες υποβάλλουν ένα ελάχιστο μέγεθος cluster στις εργασίες τους.
- HEUR [8], είναι η στρατηγική η οποία επιτρέπει μόνο στις μακριές εργασίες με υψηλό παραλληλισμό να οδηγηθούν σε rebelling.
- PRED [8], είναι η στρατηγική κατά την οποία επιλέγεται ένα μέγεθος cluster για κάθε εργασία με τον ίδιο τρόπο όπως και στη μέθοδο OPT.
- Bias [8], είναι η τάση που έχουν οι προβλέψεις να είναι συνεχώς είτε πολύ ψηλά είτε πολύ χαμηλά.

Πολιτικές ανάθεσης επεξεργαστών

- Η πολιτική equipartition [14] θέλει να επιτύχει μια δίκαιη κατανομή επεξεργαστών για όλες τις εργασίες , με τον περιορισμό ότι η αίτηση μιας διεργασίας είναι το πάνω όριο για τον αριθμό των επεξεργαστών που θα λάβει.
- Κατά τη διάρκεια μιας ανακατανομής, οι επεξεργαστές διαμοιράζονται ανάμεσα στις εργασίες με τον εξής τρόπο: κάθε εργασία ξεκινά με 0 επεξεργαστές και μετά την κατανομή κάθε εργασίας ο αριθμός αυξάνεται κατά 1 μέχρι η εργασία να ικανοποιηθεί και να αποχωρήσει από τη διαδικασία της ανάθεσης. Η διαδικασία συνεχίζεται μέχρι όλες οι εργασίες να αποσυρθούν είτε να τελειώσει ο αριθμός των διαθέσιμων επεξεργαστών.

Πολιτικές ανάθεσης επεξεργαστών

Υπάρχουν τρείς τύποι equipartitioning:

- Στη στατική equipartitioning, οι ανακατανομές δεν επιτρέπεται να αλλάξουν την κατανομή των υπαρχόντων εργασιών και έτσι μπορεί να πραγματοποιηθούν ανισορροπίες στην ανάθεση και στις ουρές.
- Στη μη συνηθισμένη equipartitioning, οι ανακατανομές συμβαίνουν μόνο στην άφιξη εργασιών και στην ολοκλήρωσή τους και για το λόγο αυτό η ανάθεση είναι δίκαιη και ακριβής όλη την ώρα, με την προϋπόθεση ότι ο επεξεργαστής επιθυμεί εργασίες που παραμένουν ίδιες κατά την εκτέλεσή τους.
- Στη δυναμική equipartitioning, οι ανακατανομές συμβαίνουν οποιαδήποτε στιγμή και η ανάθεση παραμένει δίκαιη και ακριβής.

Πολιτικές ανάθεσης επεξεργαστών

- Σε ένα πραγματικό two-level scheduling system χρησιμοποιώντας μια δυναμική πολιτική ανάθεσης επεξεργαστή, το λειτουργικό σύστημα επικοινωνεί με τον scheduler των εργασιών για να αποφασίσει την ανάθεση της παρούσα εργασίας.
- Ένας γνωστός δυναμικός αλγόριθμος ανάθεσης, είναι ο DP [14], ο οποίος αναθέτει επεξεργαστές σε πολλαπλές, προσαρμοστικές και παράλληλες εργασίες στο πρώτο επίπεδο του scheduling. Ο DP χρησιμοποιεί πληροφορίες για τις επιθυμίες του επεξεργαστή για κάθε εργασία έτσι ώστε να μπορέσει να κάνει το διαμοιρασμό δίκαια και με ακρίβεια.

Πολιτικές ανάθεσης επεξεργαστών

- Οι τρείς προϋποθέσεις για έναν καλό αλγόριθμο δυναμικής ανάθεσης επεξεργαστών είναι η δικαιοσύνη, η ακρίβεια και ο συντηρητισμός. Μαζί αυτά τα χαρακτηριστικά διαβεβαιώνουν ότι η ανάθεση θα οδηγήσει σε χαμηλό χρόνο απόκρισης και υψηλό throughput.
- Ο DP παρουσιάζει μια δυναμική εκδοχή της βασικής equipartition πολιτικής.

Παράδειγμα αλγορίθμου DP [14]

- Υποθέτουμε ότι αρχικά δε υπάρχουν εργασίες στο σύστημα και ότι έρχονται μια τη φορά. Το δίκαιο μερίδιο κάθε εργασίας δίνεται από τον παρακάτω τύπο:

$$\texttt{fair\_share} = \frac{P - \sum_{S=\{j|d_j<\lfloor P/J \rfloor\}} a_j}{J - |S|} \qquad (5.1)$$

- Όπου P είναι ο συνολικός αριθμός των επεξεργαστών και J ο αριθμός των εργασιών του συστήματος.

Παράδειγμα αλγορίθμου DP [14]

Current Allocation State	Input Event	Ending Allocation State
$\{\}$	Arrival: job 1 with $d_1 = 4$	$\{4\}$
$\{4\}$	Arrival: job 2 with $d_2 = 16$	$\{4, 12\}$
$\{4, 12\}$	Arrival: job 3 with $d_3 = 2$	$\{4, 10, 2\}$
$\{4, 10, 2\}$	Change in desire: $d_3 = 16$	$\{4, 6, 6\}$
$\{4, 6, 6\}$	Arrival: job 4 with $d_4 = 8$	$\{4, 4, 4, 4\}$
$\{4, 4, 4, 4\}$	Arrival: job 5 with $d_5 = 8$	$\{3, 3, 3, 3, 4\}$
$\{3, 3, 3, 3, 4\}$	Arrival: job 6 with $d_6 = 8$	$\{2, 2, 3, 3, 3, 3\}$
$\{2, 2, 3, 3, 3, 3\}$	Completion: job 2	$\{3, 4, 3, 3, 3\}$
$\{3, 4, 3, 3, 3\}$	Completion: job 3	$\{4, 4, 4, 4\}$
$\{4, 4, 4, 4\}$	Completion: job 6	$\{4, 6, 6\}$

Figure 5-2: A sample allocation trace for algorithm DP on a 16-processor system.

Παράδειγμα αλγορίθμου DP [14]

1. Έστω *free_procs* ο αριθμός των ελεύθερων επεξεργαστών στο σύστημα όταν φτάνει η j. Εάν *free_procs* $\geq d_j$, θέτουμε $a_j=d_j$. Διαφορετικά, θέτουμε $a_j=$ *free_procs* και προχωράμε στο βήμα 2.
2. Υπολογίζουμε την τιμή του *fair_share* που δίνεται από την εξίσωση (5.1) (συμπεριλαμβάνουμε και την εργασία j στον υπολογισμό). Εάν $a_j \leq min(d_j,$ *fair_share*), απομακρύνουμε έναν επεξεργαστή από μια εργασία που έχει την υψηλότερη ανάθεση και δίνουμε αυτόν τον επεξεργαστή στη j.
3. Επαναλαμβάνουμε το βήμα 2 μέχρι, είτε να προκύψει $a_j=d_j$, δηλαδή η j έχει ικανοποιηθεί, είτε αν $a_j=$*fair_share*, που σημαίνει πως η j πλέον έχει το δίκαιο μερίδιο επεξεργαστών της.

- Αυτά είναι τα βήματα που πραγματοποιούνται από τον DP όταν αυξάνεται η επιθυμία ενός επεξεργαστή για μια υπάρχουσα διεργασία.

Παράδειγμα αλγορίθμου DP

1. Προσθέτουμε τον αριθμό των ελεύθερων επεξεργαστών στο *free_procs*. Επαναϋπολογίζουμε το *fair_share* χρησιμοποιώντας την εξίσωση (5.1), εάν η j έχει ολοκληρωθεί.
2. Προσθέτουμε έναν επεξεργαστή σε μια στερημένη εργασία που έχει τη χαμηλότερη ανάθεση.
3. Επαναλαμβάνουμε το βήμα 2 μέχρι, είτε να ισχύει *free_procs=0* ή να μην υπάρχουν στερημένες εργασίες.

- Η εικόνα Figure 5.2 δείχνει ένα δείγμα ανάθεσης του αλγορίθμου DP σε ένα σύστημα με 16 επεξεργαστές.
- Κάθε γραμμή του πίνακα δείχνει την παρούσα κατάσταση της ανάθεσης, ένα γεγονός εισαγωγής και τη τελική φάση της ανάθεσης.

Πολιτικές ανάθεσης επεξεργαστών

- Αρκετές μελέτες έχουν δείξει πως ο αλγόριθμος DP επικρατεί σε όλες τις space-slicing πολιτικές όταν οι ανακατανομές που παρουσιάζονται είναι λίγες, παρά το φόρτο εργασίας.
- Πολλές έρευνες επικεντρώνονται στις μηχανές διαμοιρασμού μνήμης (UMA) όπου η ανάθεση των επεξεργαστών αποσυνδέεται από την ανάθεση της μνήμης.
- Άλλες πάλι έχουν δείξει ότι ο DP αποδίδει καλύτερα από τη στατική ανάθεση πολιτικής.

Πολιτικές ανάθεσης επεξεργαστών

- Γενικά, τα πλεονεκτήματα της χρήσης του αλγορίθμου DP αυξάνονται για μεγαλύτερες και πιο γρήγορες αλλαγές στο παραλληλισμό όπως επίσης και με την αύξηση του φόρτου του συστήματος. Η απόδοσή του μειώνεται όταν ο αριθμός των εργασιών είναι μεγαλύτερος από τον αριθμό των επεξεργαστών, διότι κάποιες εργασίες θα πρέπει να μπουν στη σειρά.
- Η ένταξη σε σειρές δεν είναι πρόβλημα σε μεγάλες παράλληλες μηχανές. Βέβαια, διάφορες τεχνικές έχουν υλοποιηθεί έτσι ώστε να μειώσουν την επίδραση που έχει στο χρόνο απόκρισης των εργασιών.

Πολιτικές ανάθεσης επεξεργαστών

- Οι περισσότερες έρευνες σχετικά με τις προσαρμοστικές πολιτικές διαχωρισμού για τον προγραμματισμό παράλληλων εργασιών σε παράλληλους υπολογιστές διαμοιραζόμενης μνήμης αγνοούν τους περιορισμούς που προβάλλονται από τις απαιτήσεις μνήμης των εργασιών. Αυτοί οι περιορισμοί μπορούν να έχουν αρνητική επίδραση στην απόδοση των πολιτικών διαμοιρασμού.
- Τα τελευταία χρόνια, έχουν προταθεί αρκετές προσαρμοστικές στρατηγικές διαχωρισμού για τον προγραμματισμό παράλληλων εργασιών σε message-passing multicomputers [6].

Πολιτικές ανάθεσης επεξεργαστών

- Ένα χαρακτηριστικό κλειδί των πολιτικών αυτών είναι ότι μειώνουν τον αριθμό των επεξεργαστών που αναθέτονται στις διάφορες εργασίες όσο αυξάνεται ο φόρτος του συστήματος. Οι παράλληλες εφαρμογές τυπικά αντιμετωπίζουν μια μικρή επιβράδυνση στο speedup, καθώς ο αριθμός των επεξεργαστών που αναθέτονται σε αυτές αυξάνεται.
- Σε ένα πολυπρογραμματιστικό περιβάλλον, μειώνοντας των αριθμό των επεξεργαστών που αναθέτονται σε μια εργασία, το αποτέλεσμα είναι να αυξάνεται η ακρίβεια για την εφαρμογή αυτή και επίσης να απελευθερώνονται επεξεργαστές για χρήση από άλλες εργασίες.
- Με το να αναθέτονται μικρότερα μέρη των επεξεργαστών του συστήματος σε εργασίες κατά τη διάρκεια περιόδων όπου ο φόρτος εργασίας είναι υψηλός και μεγαλύτερα μέρη όταν ο φόρτος είναι χαμηλός, οι προσαρμοστικές πολιτικές διαχωρισμού μπορούν να ξεπεράσουν πολιτικές που χρησιμοποιούν σταθερού μεγέθους τμήματα.

Πολιτικές ανάθεσης επεξεργαστών

- Στους message-passing multicomputers, όπως ο Intel Paragon, το ποσοστό της μνήμης που είναι διαθέσιμο σε εργασίες εξαρτάται από τον αριθμό των επεξεργαστών που αναθέτονται σε αυτήν.
- Ο αριθμός των επεξεργαστών που αναθέτονται σε μια εργασία πρέπει να είναι αρκετά μεγάλος ώστε τα δεδομένα και τα τμήματα του κώδικά μπορούν να χωράνε στις τοπικές θέσεις μνήμης των επεξεργαστών.
- Το κύριο κίνητρο για την παροχή της εικονικής μνήμης στους πολυυπολογιστές είναι η άνεση που προσφέρεται στον προγραμματιστή και όχι στον scheduler του συστήματος.

Πολιτικές ανάθεσης επεξεργαστών

- Υπάρχουν δύο τύποι προγραμματιστικών πολιτικών κάτω από διάφορες workload καταστάσεις. Η πρώτη πολιτική, η οποία λέγεται Adaptive Partitioning with Memory Constraints (APMC) [6] πάντα αναθέτει τουλάχιστον όσους επεξεργαστές χρειάζεται για να χωρέσει ο κώδικας και τα δεδομένα της στη μνήμη. Κατά την δεύτερη πολιτική, η οποία λέγεται Adaptive Partitioning with Virtual Memory (APVM) [6], ο αριθμός των επεξεργαστών δεν είναι τόσο απαιτητικός.
- Στρατηγικές που έχουν να κάνουν με τον προγραμματισμό των εργασιών για παράλληλους υπολογιστές ερευνούνται αρκετά τα τελευταία χρόνια.

Πολιτικές ανάθεσης επεξεργαστών

- Οι πολιτικές APMC και APVM έχουν κοινά στοιχεία με τις πολιτικές διαμοιρασμού χώρου αλλά και με τις πολιτικές gang scheduling, όταν χρησιμοποιούνται σε συστήματα όπως το Intel Paragon.
- Μοιάζουν στις πολιτικές gang scheduling καθώς είναι quantum-based πολιτικές, στις οποίες όλοι οι επεξεργαστές του συστήματος πραγματοποιούν context-switch ταυτόχρονα στο τέλος του κάθε quantum.
- Από την άλλη μεριά μοιάζουν στις πολιτικές διαμοιρασμού χώρου, στο ότι οι επεξεργαστές του συστήματος χωρίζονται σε μη επικαλυπτόμενα μέρη, τα οποία είναι αφιερωμένα σε ξεχωριστές διαδικασίες για το μέγεθος ενός quantum.

Πολιτικές ανάθεσης επεξεργαστών

- Πιο συγκεκριμένα, κατηγοριοποιούνται σαν προσαρμοστικές πολιτικές διαχωρισμού διότι κάθε εργασία που φτάνει στο σύστημα συγκαταλέγεται έτσι ώστε να εκτελεστεί σε ένα συγκεκριμένου μεγέθους partition τη στιγμή που αποκόπτεται.
- Η απόφαση αυτή βασίζεται στις απαιτήσεις της κάθε εργασίας, όπως επίσης και στις παρούσες συνθήκες φόρτου του συστήματος.
- Έχει δειχθεί ότι για να αποδώσουν καλά οι προσαρμοστικές πολιτικές διαχωρισμού στα workloads με μεγάλη μεταβλητότητα στην αίτηση εργασιών, θα πρέπει είτε οι πολιτικές να είναι παρεμποδιστικές από τη φύση τους, είτε θα πρέπει να χρησιμοποιούν πληροφορίες που πηγάζουν από το χρήστη σχετικά με την αίτηση των εργασιών.

Πολιτικές ανάθεσης επεξεργαστών

- Κατά τις πολιτικές APMC και APVM , μια εργασία χρησιμοποιεί όλη τη διαθέσιμη μνήμη των επεξεργαστών στο διαχωρισμό.
- Συγκεκριμένα κατά την πολιτική APMC, όταν μια εργασία αποκόπτεται , τα δεδομένα και ο κώδικάς της θα πρέπει να μεταφερθούν σε τοπικές μνήμες των επεξεργαστών του διαμοιρασμού της.
- Παρόμοια, όταν υπάρχει παρεμπόδιση, τα παρόντα περιεχόμενα της τοπικής μνήμης θα πρέπει να μεταφερθούν στο δίσκο.

Πολιτικές ανάθεσης επεξεργαστών

- Κατά την πολιτική APVM, το σύστημα παρακολουθεί τις ενεργές σελίδες της μνήμης και τις φορτώνει μόνο όταν μια εργασία αποκόπτεται.
- Επίσης το μέγεθος του χρονικού διαστήματος που της αντιστοιχεί θα πρέπει να είναι αρκετά μεγάλο έτσι ώστε να ξεπληρώνει τα χρέη της.

Distributed Leak algorithm

- Ο αλγόριθμος διαρροής (Leak algorithm) [4] αναπτύχθηκε για συνεχόμενη κατανομή σε συστήματα διασυνδεδεμένα με δυσδιάστατους τόρους και βασίζεται στην αρχή της διαρροής νερού.
- Από ένα σημείο προέλευσης, μια ποσότητα νερού διαρρέει προς τις κατευθύνσεις όπου δεν συναντάται αντίσταση. Ένα σημαντικό στοιχείο που πρέπει να θυμόμαστε είναι ότι το διαρρέον νερό παρουσιάζει συνοχή, η οποία περιορίζει τη διάμετρο της προκύπτουσας λίμνης.

Distributed Leak algorithm

- Για μια κατανεμημένη ανάθεση, ο αριθμός των επεξεργαστών που θα κατανεμηθούν αντιστοιχεί στο ποσό της διαρροής νερού. Οι επεξεργαστές που έχουν ήδη κατανεμηθεί στο πλέγμα (mesh) είναι οι περιοχές αντίστασης και η τελική περιοχή που σχηματίζεται από τους κατανεμημένους επεξεργαστές είναι η προκύπτουσα λίμνη.

Distributed Leak algorithm [4]


```
WHILE (true) DO
   wait for search_initial_message(load, requester)
   IF (node is free)
      initial = myself
      send it allocate_message(initial, load) to myself // starts allocation
      wait for confirmation_message(remaining_load)
      IF (remaining_load >0 )
         IF (not last node in sequence)
            send search_initial_message(remaining_load, requester) to next in sequence
            // increase/decrease column or/and increase/decrease line depending on origin point
         ELSE
            send(`allocation failed`) to requester
            send release_message() to node_list
         ENDIF
      ELSE
         send(`allocation succeeded`) to requester
      ENDIF
   ELSE
      IF (not last node in sequence)
         send search_initial_message(remaining_load, requester) to next in sequence
      ELSE
         send(`allocation failed`) to requester
         send release_message() to node_list
      ENDIF
   ENDIF
ENDWHILE
```

Fig. 2. Phase 1 of an initial allocation: search for initial node.

Distributed Leak algorithm [4]


```
WHILE (true) DO
   wait for allocate_message(initial, load)
   set node to occupied and add to node_list
   decrement load
   IF (load == 0)
      send confirmation_message(load) to initial
   ELSE
      no_of_free_neighbors = check for free neighbors()
      avg_load = load / no_of_free_neighbors
      send allocate_message(myself, avg_load) to all free neighbors
      wait for confirmations // 0 if succeeded, any other number is remaining load
      accumulate remaining_load and combine node_lists
      send confirmation_message(remaining_load, node_list) to initial
   ENDIF
ENDWHILE
```

Fig. 3. Phase 2 of an initial allocation: recursive distributed allocation.

Distributed Leak algorithm

- Στον αλγόριθμο αυτό, μια αρχική κατανομή αποτελείται από δύο φάσεις.
- Στην πρώτη φάση (fig. 2) οι κόμβοι μηχανής αναζητούνται από το σημείο προέλευσης χρησιμοποιώντας ένα διαδοχικό κύμα αναζήτησης (στη χρησιμοποιούμενη τοπολογία πλέγματος οι κόμβοι αναζητούνται με ένα μοτίβο «φιδιού» μέχρι όλες οι γραμμές να μετατοπιστούνε και οι τέσσερις γωνίες να είναι εφικτές μονάδες προέλευσης).
- Δεδομένου ότι αυτή η λειτουργία προέρχεται έξω από το μηχάνημα, αναφερόμαστε στα σημεία προέλευσης ως σημεία εισόδου. Ανάλογα με τη θέση του σημείου εισόδου ο προσανατολισμός του μοτίβου αναζήτησης είναι προσαρμοσμένος (αριστερά-δεξιά, πάνω-κάτω).

Distributed Leak algorithm

- Στη δεύτερη φάση (fig. 3) όλοι οι άμεσοι γείτονες του σημείου προέλευσης δοκιμάζονται παράλληλα για να καθοριστεί εάν υπάρχουν ελεύθεροι.
- Κάθε ελεύθερος γείτονας τότε γίνεται μέρος του φόρτου και η δεύτερη φάση συνεχίζεται αναδρομικά και παράλληλα μέχρι να μην υπάρχει άλλος διαθέσιμος φόρτος.
- Όλοι οι κόμβοι που είναι ελεύθεροι δοκιμάζονται ως σημεία προέλευσης μέχρι ένα ελεύθερο partition του κατάλληλου μεγέθους να βρεθεί, ή μην είναι διαθέσιμοι άλλοι κόμβοι για να δοκιμάσουν, και η κατανομή απορρίπτεται.

Distributed Leak algorithm [4]

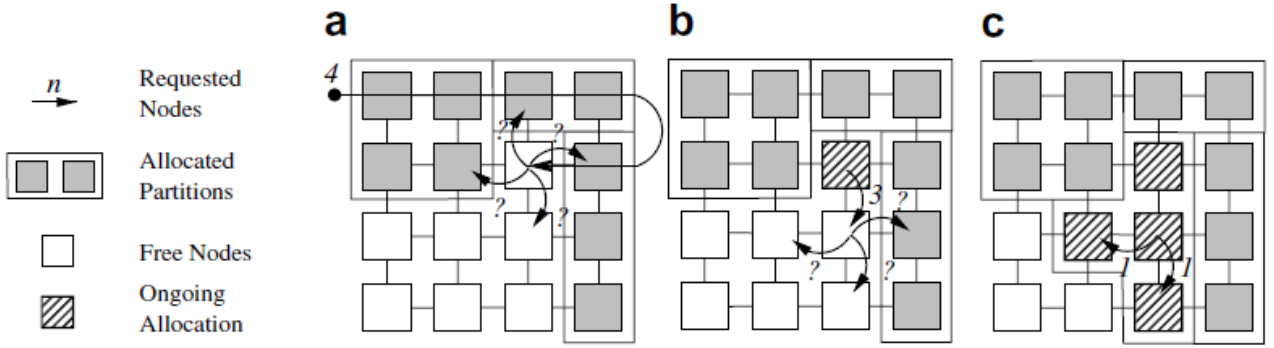

Fig. 4. Contiguous Leak algorithm.

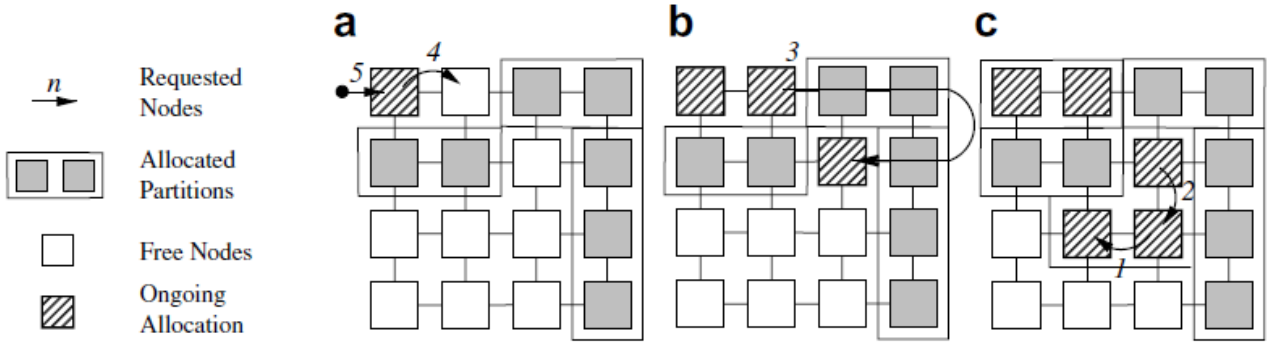

Fig. 5. Noncontiguous Leak algorithm.

Συμπερασματικά

- Υπάρχουν πολλές πολιτικές αλλά και αλγόριθμοι που καθορίζουν τον τρόπο ανάθεσης των επεξεργαστών στα παράλληλα συστήματα.
- Η ανάθεση αυτή πραγματοποιείται βασιζόμενη σε πολλές παραμέτρους που επηρεάζουν την απόδοση του συστήματος είτε θετικά είτε αρνητικά και για το λόγο αυτό δεν έχει επικρατήσει ακόμα, αποκλειστικά μία πολιτική.
- Πολλαπλές στρατηγικές έχουν προταθεί την τελευταία δεκαετία και συνεχίζουν να αναπτύσσονται προσπαθώντας να ανταποκριθούν στις εξελίξεις και να οδηγήσουν στη βέλτιστη λύση.

Βιβλιογραφία

1. Shikharesh Majumdar, Derek L. Eager and Richard B. Bunt (1990), Characterisation of programs for scheduling in multiprogrammed parallel systems. *Department of Computational Science, University of Saskatchewan, Saskatoon. Canada*
2. K.C. Sevcik (1994), Application scheduling and processor allocation in multiprogrammed parallel processing systems. *Computer Systems Research Institute, University of Toronto*
3. S. Bani-Mohammad, M. Ould-Khaoua and I. Ababneh (2007), An efficient non-contiguous processor allocation strategy for 2D mesh connected multicomputers. *Department of Computing Science, University of Glasgow, Department of Computing Science, Al al-Bayt University*
4. Ce´sar A.F. De Rose, Hans-Ulrich Heiss and Barry Linnert (2007), Distributed dynamic processor allocation for multicomputers. *Catholic University of Rio Grande do Sul (PUCRS), Department of Computer Science,* Technical University Berlin, Faculty of Electrical Engineering and Computer Science
5. Anca I.D. Bucur and Dick H.J. Epema (2002), Local versus Global Schedulers with Processor Co-allocation in Multicluster Systems. *Faculty of Information Technology and Systems, Delft University of Technology*
6. Sanjeev K. Setia (1996), The Interaction between Memory Allocation and Adaptive Partitioning in Message-Passing Multicomputers. *Department of Computer Science, George Mason University*
7. Kelvin K. Yue and David J. Lilja (1996), Loop-Level Process Control: An Effective Processor Allocation Policy for Multiprogrammed Shared-Memory Multiprocessors. *Department of Computer Science and Department of Electrical Engineering,University of Minnesota*

Βιβλιογραφία

8. Allen B. Downey (1998), Using Queue Time Predictions for Processor Allocation. *University of California*
9. S. Banen, A.I.D. Bucur, and D.H.J. Epema (2003), A Measurement-Based Simulation Study of Processor Co-allocation in Multicluster Systems. *Faculty of Information Technology and Systems Delft University of Technology*
10. Dror G. Feitelson, Larry Rudolph, and Uwe Schwiegelshohn (2005), Parallel Job Scheduling — A Status Report. *The Hebrew University of Jerusalem, Massachusetts Institute of Technology and University of Dortmund*
11. Fang Wang, Hubertus Franker, Marios Papaefthymiou, Pratap Pattnalk, Larry Rudolph and Mark S. Squillante (1997), A Gang Scheduling Design for Multiprogrammed Parallel Computing Environments. *Computer Science Department, Yale University and IBM Research Division, T.J. Watson Research Center*
12. Jitendra Padhye and Lawrence Dowdy (1997), Dynamic versus Adaptive Processor Allocation Policies for Message Passing Parallel Computers: An Empirical Comparison. *Department of Computer Science, University of Massachusetts at Amherst and Department of Computer Science, Vanderbilt University*
13. Dror G. Feitelson and Larry Rudolph (1999), Metrics and Benchmarking for Parallel Job Scheduling. *Institute of Computer Science, The Hebrew University of Jerusalem and Laboratory for Computer Science, MIT*
14. Siddhartha Sen (2004), Dynamic Processor Allocation for Adaptively Parallel Work-Stealing Jobs. *Massachusetts Institute of Technology*
15. Rajesh K. Mansharamani and Mary K. Vernon (1993), Comparison of Processor Allocation Policies for Parallel Systems. *Computer Science Department, University of Wisconsin*
16. E. Smirni, E. Rosti, L. W. Dowdy and G. Serazzi (1994), Evaluation of multiprocessor allocation policies. *Vanderbilt University, Universita di Milano and Politecnico di Milano*