Υλοποιηση της τεχνικης της προωθησης σελιδων μνημης σε συσταδες υπολογιστων

Θωμάς Ρεπαντής

repantis@hpclab.ceid.upatras.gr

Διπλωματική Εργασία

Επιβλέπων: Καθηγητής **Θεόδωρος Σ. Παπαθεοδώρου**
Συνεπιβλέπων: Αναπληρωτής Καθηγητής **Δημήτριος Ν. Σερπάνος**
Πάτρα, Νοέμβριος 2002

Επισκόπηση:

1. Οι συστάδες υπολογιστών

2. Οι κατανεμημένες κοινές μνήμες λογισμικού

3. Η ανάγκη για προώθηση σελίδων μνήμης

4. Ο μηχανισμός προώθησης σελίδων μνήμης της κατανεμημένης κοινής μνήμης λογισμικού JIAJIA

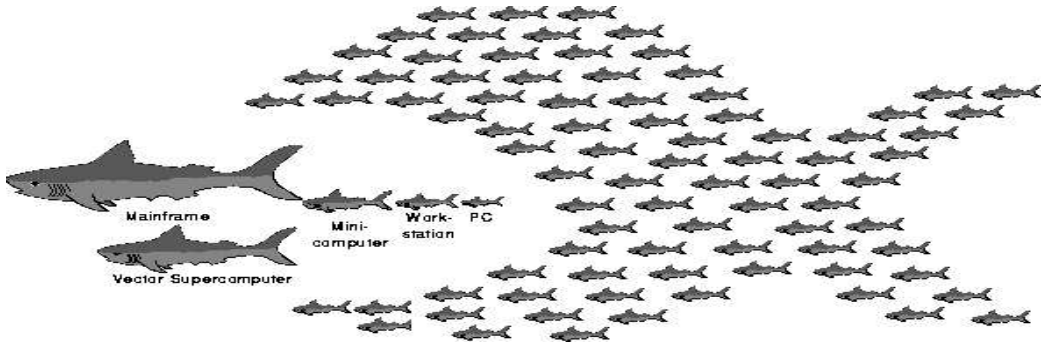
5. Η υλοποίηση του νέου μηχανισμού προώθησης σελίδων μνήμης

6. Πειράματα

7. Μετρήσεις

8. Συμπεράσματα

Οι *συστάδες* (clusters) υπολογιστών:



- Αντιμετωπίζουν απαιτητικά υπολογιστικά προβλήματα.
- Δημιουργούνται διασυνδέοντας πολλούς COTS υπολογιστές.
- Έχουν χαμηλές συγκριτικά επιδόσεις, αλλά χαμηλό κόστος, ευελιξία και παραμετροποιησιμότητα.

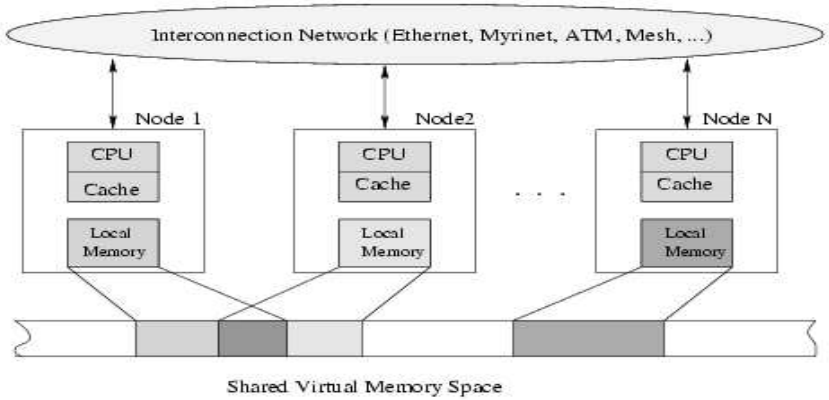
Συμβατικός διαχωρισμός των παράλληλων αρχιτεκτονικών:

- Κατανεμημένης μνήμης
- Κοινής μνήμης

Συστήματα *Κατανεμημένης Κοινής Μνήμης* (Distributed Shared Memory — DSM ή SVM):

- Χρησιμοποιούν κατανεμημένη μνήμη, αλλά ο προγραμματιστής έχει την εικόνα κοινής.
- Προγραμματιστική ευκολία, αλλά χαμηλές επιδόσεις.

Κατανεμημένες Κοινές Μνήμες υλοποιημένες με Λογισμικό:



- Ένα επίπεδο λογισμικού παρέχει την ψευδαίσθηση της κοινής μνήμης στον προγραμματιστή εφαρμογών.

- Μοντέλο συνέπειας μνήμης και πρωτόκολλο συνοχής λανθάνουσας μνήμης.

Η σημασία της προώθησης (μετανάστευσης) σελίδων μνήμης στους σταθμούς ενός SWDSM που τις χρησιμοποιούν περισσότερο:

- Προσαρμογή σε μεταβαλλόμενους πόρους (σε συνδυασμό με μετανάστευση εργασιών).
- Προσαρμογή στις ανάγκες κάθε εφαρμογής.
- Αύξηση της τοπικότητας στις προσβάσεις στην κοινή μνήμη.
- Σε ένα πρωτόκολλο βασισμένο σε έδρες οι επιδόσεις εξαρτώνται πολύ από την κατανομή των εδρών.

Το JIAJIA SWDSM:

- Απλό και αποδοτικό.
- SPMD, NUMA (home-based).
- Write Invalidate, scope consistency, lock-based.
- Οι αιτήσεις για εκτός έδρας σελίδες επιφέρουν SIGSEGV, τη μεταφορά και αποθήκευσή τους στη λανθάνουσα μνήμη (INV, RO, RW).
- Απλό API.

Ο υπάρχων αλγόριθμος μετανάστευσης σελίδων:

- Όποιος φθάνει σε φράγμα, ειδοποιεί για εγγραφές.
- Ο διαχειριστής φράγματος λαμβάνει τις αιτήσεις φράγματος και τις ειδοποιήσεις εγγραφής.
- Όταν ο διαχειριστής λάβει όλες τις αιτήσεις, εκπέμπει όλες τις ειδοποιήσεις εγγραφών.
- Στην απόκριση φράγματος κάθε σταθμός ελέγχει από τις ειδοποιήσεις εγγραφής αν κάποια σελίδα τροποποιήθηκε από ένα μόνο σταθμό. Τότε *ο μοναδικός τροποποιητής γίνεται η νέα έδρα* (ανανεώνοντας τους πίνακες σελίδων).

Ο νέος αλγόριθμος μετανάστευσης σελίδων:

- Για κάθε εντός έδρας σελίδα μετρούνται τα bytes των απομακρυσμένων διαφορών.

- Όποιος φθάνει σε φράγμα στέλνει ειδοποιήσεις εγγραφής και διευθύνσεις σελίδων με τους ισχυρούς τροποποιητές τους, εφόσον ξεπερνάται κάποιο κατώφλι.

- Ο διαχειριστής φράγματος συγκεντρώνει ειδοποιήσεις εγγραφής και πληροφορίες μετανάστευσης.

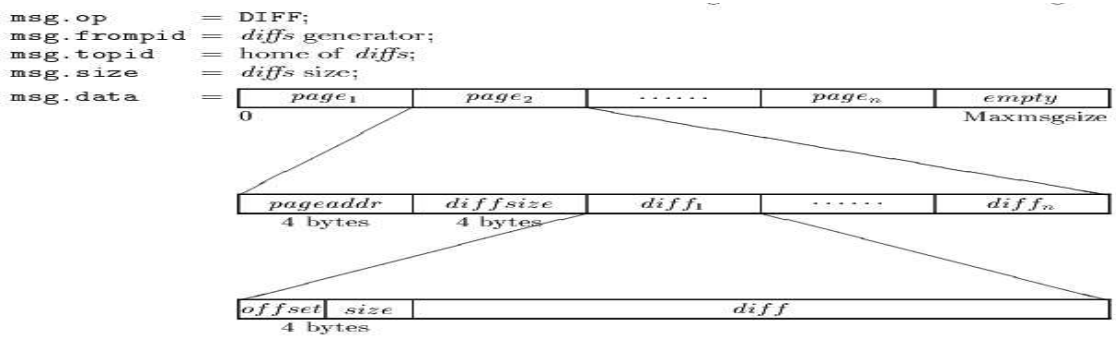
- Όταν λάβει όλες τις αιτήσεις, εκπέμπει και τα παραπάνω.

- Στην απόκριση φράγματος κάθε σταθμός ελέγχει τις πληροφορίες μετανάστευσης και για κάθε σελίδα που περιέχεται εκεί *ο ισχυρός τροποποιητής γίνεται η νέα έδρα* (ανανεώνοντας τους πίνακες σελίδων και μετακινώντας τη σελίδα αν απαιτείται).

Υλοποίηση:

- rsh $\rightsquigarrow$ ssh για λόγους ασφαλείας!

- Μέτρηση απομακρυσμένων διαφορών από τα μηνύματα.

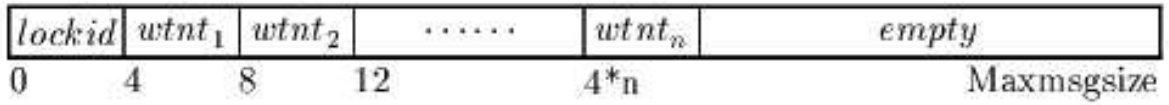


- Αθροιστική αποθήκευση του μεγέθους των απομακρυσμένων διαφορών σε ειδικό πίνακα στη δομή κάθε εντός έδρας σελίδας.

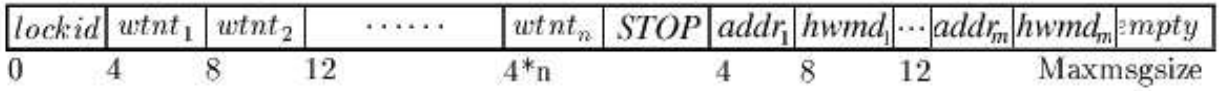
Σε κάθε φράγμα κάθε σταθμός υπολογίζει τις μέγιστες τροποποιήσεις και τους ισχυρότερους τροποποιητές των εντός έδρας σελίδων του (`migfindmax()`).

Φορτώνει τις πληροφορίες στο μήνυμα αίτησης φράγματος:

```
msg.op      = BARR;
msg.frompid = all barrier() caller;
msg.topid   = home of hidelock;
msg.size    = 4+4*n;
msg.data    =
```



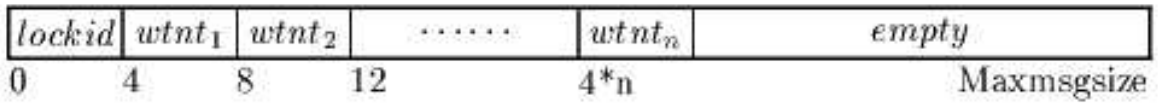

```
msg.op      = BARR;
msg.frompid = all barrier() caller;
msg.topid   = home of hidelock;
msg.size    = 4+4*n+4+4*2*m;
msg.data    =
```



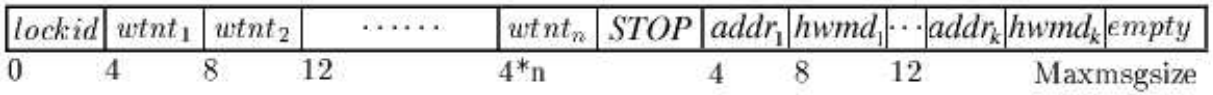
Ο διαχειριστής φράγματος συλλέγει όλες τις πληροφορίες μετανάστευσης σε ένα αλφαριθμητικό στη δομή κάθε κλειδιού.

Τις φορτώνει στο μήνυμα παραχώρησης φράγματος:

```
msg.op      = BARRGRANT;
msg.frompid = home of hidelock;
msg.topid   = all hosts;
msg.size    = 4+4*n;
msg.data    =
```




```
msg.op      = BARRGRANT;
msg.frompid = home of hidelock;
msg.topid   = all hosts;
msg.size    = 4+4*n+4+4*2*k;
msg.data    =
```



- Κάθε σταθμός εξετάζει όλες τις πληροφορίες μετανάστευσης και από τις ειδοποιήσεις εγγραφής σημειώνει ποιες από τις σελίδες που θα μεταναστεύσουν είναι έγκυρες στη λανθάνουσα μνήμη του μοναδικού τροποποιητή τους.

- Η νέα έδρα κάποιας σελίδας που δεν ανήκει σε αυτή την κατηγορία τη λαμβάνει από την έως τότε έδρα της.

- Τέλος ενημερώνονται οι πίνακες σελίδων (`home`, `cache`, `global`) σε παλαιές και νέες έδρες και σε κάθε άλλο σταθμό.

Με τη μετακίνηση σελίδων δημιουργούνται επιπλέον ανάγκες συγχρονισμού. Απαραίτητη η εξασφάλιση ότι:

- Η νέα έδρα δε θα αναφερθεί στη νεοαποκτηθείσα σελίδα, πριν αυτή έλθει πραγματικά από την παλαιά έδρα.

- Η παλαιά έδρα δε θα αλλάξει τους πίνακες σελίδων της και δε θα αποαντιστοιχίσει τη σελίδα, πριν τη στείλει στη νέα έδρα.

Δύο τρόποι εξασφάλισης:

1. Με μεταβλητές συγχρονισμού και αναδιάταξη των εισερχόμενων μηνυμάτων, ώστε οι εξυπηρέτες να μην οδηγούνται σε αδιέξοδο.

2. Με διάσπαση της `migpage()` σε `migpagesarriving()` και `migpagesleaving()`, με σημείο συγχρονισμού ανάμεσά τους.

Η δεύτερη λύση είναι πιο απλή και στιβαρή, ίσως ελάχιστα πιο χρονοβόρα. Προτιμάται.

Επιβεβαιώθηκε σε κάθε στάδιο η σωστή λειτουργία του μηχανισμού, με την παρακολούθηση της μετάδοσης πληροφοριών μετανάστευσης και σελίδων.

Υλοποιήθηκαν δύο μικρομετροπρογράμματα:

1. Πρόκλησης μετανάστευσης σελίδων λόγω αποκλειστικών απομακρυσμένων εγγραφών.

2. Πρόκλησης μετανάστευσης σελίδων λόγω ισχυρότερου τροποποιητή.

Εκτελώντας τα μικρομετροπρογράμματα επιβεβαιώθηκε ότι ο μηχανισμός μεταναστεύει τις σελίδες που πρέπει.

Πλατφόρμες εκτέλεσης μετρήσεων:

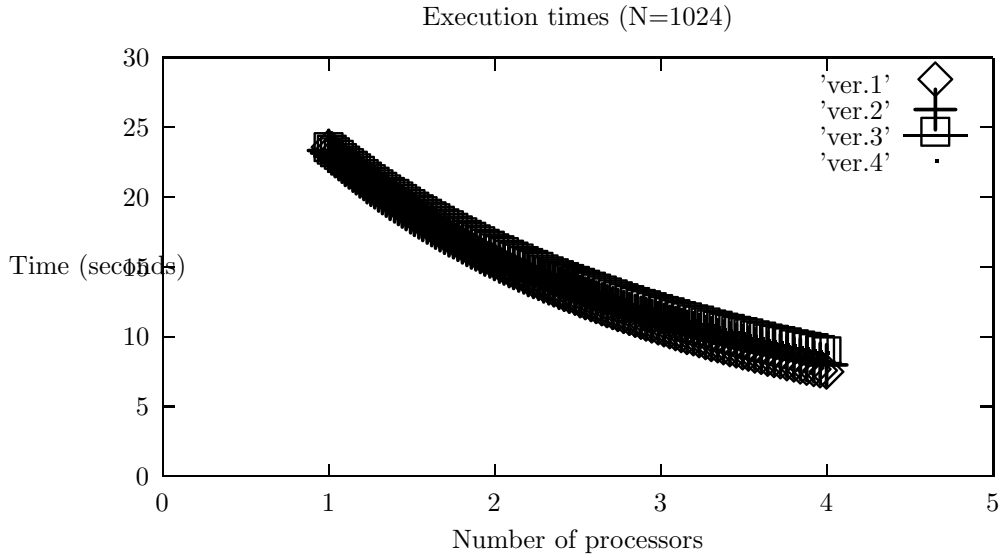
1. 4 Intel Pentium III @ 866MHz, 256KB, 256MB, GigaBit Ethernet, Linux, gcc

2. 2 Intel Pentium @ 133MHz 64MB, 100MB Ethernet, Linux, gcc

Εφαρμογές:

Water, LU, EP, IS, SOR, TSP, PI, MM

Αξιολόγηση παράλληλου πολλαπλασιασμού (MM) στο JIAJIA



Καλή χρονοβελτίωση και κλιμακωσιμότητα.

Μετρήθηκε ο αριθμός των σελίδων που μεταβάλλονται από σταθμούς πλην της έδρας τους, αλλά δεν έχουν μοναδικό τροποποιητή:

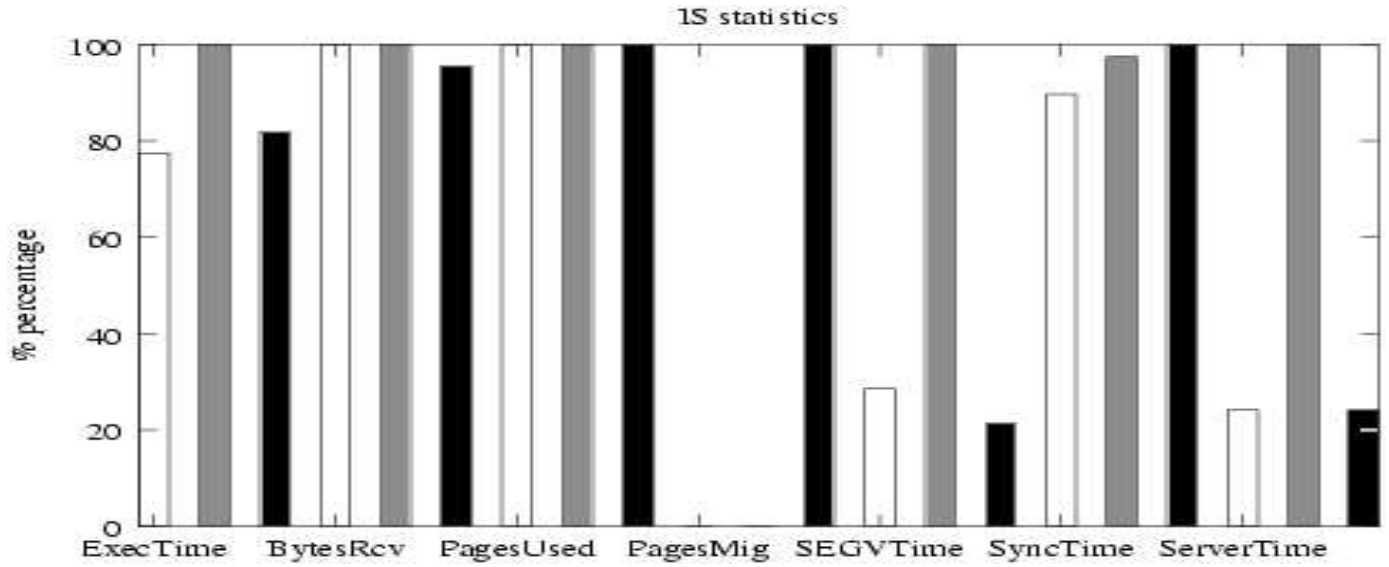
Υπάρχουν περιθώρια βελτίωσης του υπάρχοντος πρωτοκόλλου μετανάστευσης σελίδων (ιδίως σε SOR, TSP, WATER).

Παράμετροι βελτιστοποίησης απόδοσης:

1. Το κατώφλι μετανάστευσης (εμπειρική εύρεση).
   (`jia_config(HMIGthreshold, int value)`).

2. Η ακριβής θέση ενεργοποίησης του μηχανισμού μετανάστευσης.
   (`jia_config(HMIG, ON/OFF)`).

Σύγκριση των τριων περιπτώσεων (καμμία μετανάστευση, υπάρχων πρωτόκολλο, νέο πρωτόκολλο) για μια τυπική εφαρμογή σε 4 σταθμούς.



Ενθαρρυντικά αποτελέσματα: Μικρός χρόνος εκτέλεσης, αποφυγή SIGSEGVs, εξοικονόμηση μεταδιδόμενης πληροφορίας.

Οι διαθέσιμες εφαρμογές δεν προσφέρονταν για την ανάδειξη των πλεονεκτημάτων της μετανάστευσης σελίδων:

- Κάνουν χρήση λίγων σελίδων κοινής μνήμης, άρα λίγες σελίδες μεταναστεύουν.

- Περιορισμένες αναφορές σε απομακρυσμένες σελίδες, άρα μικρή μεταφερόμενη πληροφορία.

- Μικροί χρόνοι εκτέλεσης, άρα μεγάλο σχετικό πειραματικό σφάλμα.

- Δεν έχουν όλες συχνά φράγματα, άρα λίγες ευκαιρίες για ενεργοποίηση μηχανισμού.

Μελλοντική εργασία:

- Βελτίωση υλοποίησης της τεχνικής (άλλες πολιτικές, ανίχνευση παλινδρομήσεων σελίδων, μέτρηση τοπικών προσβάσεων, αυτόματη εύρεση κατωφλίου και σημείων κλήσεων μετανάστευσης στον κώδικα).

- Ανάδειξη προτερημάτων τεχνικής (δοκιμές με εφαρμογές με μεγαλύτερη ανάγκη για μετανάστευση σελίδων, δοκιμές με μεγαλύτερα μεγέθη προβλημάτων)