Βελτιώνοντας την επίδοση του packrat parsing

Νίκος Μαυρογεώργης

Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Εθνικό Μετσόβειο Πολυτεχνείο

Παρουσίαση Διπλωματικής
Ιούνιος 2020
Επιβλέπων Καθηγητής: Νίκος Παπασπύρου

Πίνακας Περιεχομένων

Πίνακας Περιεχομένων

Ιδέα

- Θέλουμε να βρούμε κάποιο κομμάτι του αλγορίθμου που μπορεί να εκτελεστεί από πολλά νήματα ταυτόχρονα, ώστε να μοιραστεί ο φόρτος εργασίας

Ιδέα

- Θέλουμε να βρούμε κάποιο κομμάτι του αλγορίθμου που μπορεί να εκτελεστεί από πολλά νήματα ταυτόχρονα, ώστε να μοιραστεί ο φόρτος εργασίας
- Στη διατεταγμένη επιλογή $E \leftarrow E_1/E_2/E_3$ οι υποεκφράσεις αναλύονται ξεκινώντας από το ίδιο σημείο της εισόδου

Ιδέα

- Θέλουμε να βρούμε κάποιο κομμάτι του αλγορίθμου που μπορεί να εκτελεστεί από πολλά νήματα ταυτόχρονα, ώστε να μοιραστεί ο φόρτος εργασίας
- Στη διατεταγμένη επιλογή $E \leftarrow E_1/E_2/E_3$ οι υποεκφράσεις αναλύονται ξεκινώντας από το ίδιο σημείο της εισόδου
- Αν αποτύχει η E_1, τότε ξαναπροσπαθεί από το ίδιο σημείο η E_2 κλπ

Ιδέα

- Θέλουμε να βρούμε κάποιο κομμάτι του αλγορίθμου που μπορεί να εκτελεστεί από πολλά νήματα ταυτόχρονα, ώστε να μοιραστεί ο φόρτος εργασίας
- Στη διατεταγμένη επιλογή $E \leftarrow E_1/E_2/E_3$ οι υποεκφράσεις αναλύονται ξεκινώντας από το ίδιο σημείο της εισόδου
- Αν αποτύχει η E_1, τότε ξαναπροσπαθεί από το ίδιο σημείο η E_2 κλπ
- Θεωρητικά θα μπορούσαμε να τις αναθέσουμε σε ξεχωριστά νήματα

Ιδέα

- Θέλουμε να βρούμε κάποιο κομμάτι του αλγορίθμου που μπορεί να εκτελεστεί από πολλά νήματα ταυτόχρονα, ώστε να μοιραστεί ο φόρτος εργασίας
- Στη διατεταγμένη επιλογή $E \leftarrow E_1/E_2/E_3$ οι υποεκφράσεις αναλύονται ξεκινώντας από το ίδιο σημείο της εισόδου
- Αν αποτύχει η E_1, τότε ξαναπροσπαθεί από το ίδιο σημείο η E_2 κλπ
- Θεωρητικά θα μπορούσαμε να τις αναθέσουμε σε ξεχωριστά νήματα
- Αναδρομική κλήση νημάτων: Αν κάποιο νήμα που κληθεί κατά την διατεταγμένη επιλογή πάει να αναλύσει με τη σειρά του διατεταγμένη επιλογή, καλεί και δικά του νήματα

Λεπτά σημεία

- Τα ενδιάμεσα αποτελέσματα πρέπει να προστατεύονται από ταυτόχρονη χρήση

Λεπτά σημεία

- Τα ενδιάμεσα αποτελέσματα πρέπει να προστατεύονται από ταυτόχρονη χρήση
- Αν το νήμα που ανέλυσε την E_2, επιστρέψει επιτυχώς, πρέπει πάλι να περιμένουμε το νήμα που αναλύει την E_1, καθώς η υποέκφραση αυτή έχει προτεραιότητα

Λεπτά σημεία

- Τα ενδιάμεσα αποτελέσματα πρέπει να προστατεύονται από ταυτόχρονη χρήση
- Αν το νήμα που ανέλυσε την E_2, επιστρέψει επιτυχώς, πρέπει πάλι να περιμένουμε το νήμα που αναλύει την E_1, καθώς η υποέκφραση αυτή έχει προτεραιότητα
- Αν το νήμα που ανέλυσε την E_i, επιστρέψει επιτυχώς, πρέπει τερματίσουμε τα νήματα που ανέλαβαν τις $E_{i+1}, \ldots, E_n$

Λεπτά σημεία

- Τα ενδιάμεσα αποτελέσματα πρέπει να προστατεύονται από ταυτόχρονη χρήση
- Αν το νήμα που ανέλυσε την E_2, επιστρέψει επιτυχώς, πρέπει πάλι να περιμένουμε το νήμα που αναλύει την E_1, καθώς η υποέκφραση αυτή έχει προτεραιότητα
- Αν το νήμα που ανέλυσε την E_i, επιστρέψει επιτυχώς, πρέπει τερματίσουμε τα νήματα που ανέλαβαν τις $E_{i+1}, \ldots, E_n$
- Αν υπάρχουν πολλές εναλλακτικές στην επιλογή, τότε ίσως να μην αξίζει να καλέσουμε πολλά νήματα, καθώς το overhead του fork-join θα είναι μεγάλο

Λεπτά σημεία

- Τα ενδιάμεσα αποτελέσματα πρέπει να προστατεύονται από ταυτόχρονη χρήση
- Αν το νήμα που ανέλυσε την E_2, επιστρέψει επιτυχώς, πρέπει πάλι να περιμένουμε το νήμα που αναλύει την E_1, καθώς η υποέκφραση αυτή έχει προτεραιότητα
- Αν το νήμα που ανέλυσε την E_i, επιστρέψει επιτυχώς, πρέπει τερματίσουμε τα νήματα που ανέλαβαν τις $E_{i+1}, \ldots, E_n$
- Αν υπάρχουν πολλές εναλλακτικές στην επιλογή, τότε ίσως να μην αξίζει να καλέσουμε πολλά νήματα, καθώς το overhead του fork-join θα είναι μεγάλο
- Αν το βάθος του fork-join tree είναι ήδη μεγάλο, τότε ίσως πάλι να μην αξίζει να καλέσουμε νέα νήματα

Λεπτά σημεία - Κλειδώματα

- Για να αποφευχθούν συνθήκες ανταγωνισμού για τα κελιά πρέπει να προστατεύονται με κλειδώματα
- Ένας απλός και αποτελεσματικός τρόπος είναι κάθε κελί να έχει το κλειδί του οπότε για να το υπολογίσει κάποιος θα πρέπει να το κλειδώσει

Λεπτά σημεία - Κλειδώματα

- Για να αποφευχθούν συνθήκες ανταγωνισμού για τα κελιά πρέπει να προστατεύονται με κλειδώματα
- Ένας απλός και αποτελεσματικός τρόπος είναι κάθε κελί να έχει το κλειδί του οπότε για να το υπολογίσει κάποιος θα πρέπει να το κλειδώσει

```
1 cur_cell->lock();
2 cur_cell->set_res(Result::pending);
3 cur_cell->unlock();
```

Λεπτά σημεία - Κλειδώματα

- Για να αποφευχθούν συνθήκες ανταγωνισμού για τα κελιά πρέπει να προστατεύονται με κλειδώματα
- Ένας απλός και αποτελεσματικός τρόπος είναι κάθε κελί να έχει το κλειδί του οπότε για να το υπολογίσει κάποιος θα πρέπει να το κλειδώσει

```
1 cur_cell->lock();
2 cur_cell->set_res(Result::pending);
3 cur_cell->unlock();
```

- Όσο το κελί είναι pending (δηλαδή το υπολογίζει κάποιος) οι υπόλοιποι κάνουν busy-wait

Λεπτά σημεία - Ιεραρχία

- Κάθε νήμα που καλείται λαμβάνει έναν αριθμό *rank* από το γονέα
- Υψηλότερο *rank* είναι το 0

Λεπτά σημεία - Ιεραρχία

- Κάθε νήμα που καλείται λαμβάνει έναν αριθμό *rank* από το γονέα
- Υψηλότερο *rank* είναι το 0

```
1  for (auto j = 0; j < i; ++j) {
2      threads[j].join();
3      if (results[j]) {
4          finished_rank.store(j);
5          pos = positions[j];
6          for (auto k = j + 1; k < i; ++k) {
7              threads[k].join();
8          }
9          return true;
10     }
11 }
12 pos = orig_pos;
13 return false;
```

Λεπτά σημεία - Τερματισμός

- Περιοδικός έλεγχος για πρόωρο τερματισμό από τον γονέα

Λεπτά σημεία - Τερματισμός

- Περιοδικός έλεγχος για πρόωρο τερματισμό από τον γονέα

```
1    auto fr = parent_finished_rank->load();
2    if (fr >= 0 && fr < rank) {
3        return false;
4    }
```

Όλα μαζί

```
finished_rank.store(-1);

int results[exprs.size()];
int positions[exprs.size()];
std::vector<std::thread> threads;

auto i = 0;
for (auto& expr : exprs) {
    threads.emplace_back([&, expr, i]()
    {
        SimpleWorker sw{in, peg, cells, pos, expr_limit,
                        cur_tree_depth + 1, max_tree_depth, i, &finished_rank};
        results[i] = expr->accept(sw);
        positions[i] = sw.cur_pos();
    });
    i++;
}
```

Πίνακας Περιεχομένων

Δοκιμαστικά προγράμματα

- 3 αρχεία από τον πηγαίο κώδικα της Java
  - *Arrays.java* - 116K
  - *BigDecimal.java* - 140K
  - *Throwable.java* - 28K
- Μέτριο προς μεγάλο μέγεθος ώστε να φανούν οι διαφορές στους χρόνους εκτέλεσης των αλγορίθμων
- 12 νήματα διαθέσιμα
- *std::chrono::high_resolution_clock* της STL

Ρύθμιση των παραμέτρων

- Τόσο για το elastic όσο και για το παράλληλο packrat εκτελούμε πειράματα για να βελτιώσουμε τους συνδυασμούς των παραμέτρων

Ρύθμιση των παραμέτρων

- Τόσο για το elastic όσο και για το παράλληλο packrat εκτελούμε πειράματα για να βελτιώσουμε τους συνδυασμούς των παραμέτρων
- Για το elastic δοκιμάζουμε παράθυρα μήκους $256, 512, 1024$ και κατώφλια απενεργοποίησης $0, 16, 32, 48$

Ρύθμιση των παραμέτρων

- Τόσο για το elastic όσο και για το παράλληλο packrat εκτελούμε πειράματα για να βελτιώσουμε τους συνδυασμούς των παραμέτρων
- Για το elastic δοκιμάζουμε παράθυρα μήκους $256, 512, 1024$ και κατώφλια απενεργοποίησης $0, 16, 32, 48$

Καλύτερος Συνδυασμός

$w = 256, \mathit{thres} = 32$

Ρύθμιση των παραμέτρων

- Τόσο για το elastic όσο και για το παράλληλο packrat εκτελούμε πειράματα για να βελτιώσουμε τους συνδυασμούς των παραμέτρων
- Για το elastic δοκιμάζουμε παράθυρα μήκους $256, 512, 1024$ και κατώφλια απενεργοποίησης $0, 16, 32, 48$

Καλύτερος Συνδυασμός

$w = 256, \mathit{thres} = 32$

- Για τον παράλληλο δοκιμάζουμε όρια μήκους διατεταγμένης επιλογής $2, 4, 6, 8$ και και μέγιστο βάθος δέντρου $0, 1, 2$

Ρύθμιση των παραμέτρων

- Τόσο για το elastic όσο και για το παράλληλο packrat εκτελούμε πειράματα για να βελτιώσουμε τους συνδυασμούς των παραμέτρων
- Για το elastic δοκιμάζουμε παράθυρα μήκους $256, 512, 1024$ και κατώφλια απενεργοποίησης $0, 16, 32, 48$

Καλύτερος Συνδυασμός

$w = 256, \mathit{thres} = 32$

- Για τον παράλληλο δοκιμάζουμε όρια μήκους διατεταγμένης επιλογής $2, 4, 6, 8$ και και μέγιστο βάθος δέντρου $0, 1, 2$

Καλύτερος Συνδυασμός

$\mathit{expr\_limit} = 4, \mathit{max\_depth} = 1$. Δηλαδή 5 ενεργά νήματα κατά το μέγιστο.

Τελική Σύγκριση

Χρόνοι εκτέλεσης (ms)			
Αλγόριθμος Packrat	Arrays 116K	BigDecimal 140K	Throwable 28K
Κλασικός	404	350	46
Elastic (256, 32)	380	329	51
Παράλληλος (1, 4)	432	369	62

Πίνακας: Τελικά αποτελέσματα για τους τρεις αλγορίθμους

Πίνακας Περιεχομένων

Συμπεράσματα

- Αδιαφιλονίκητος νικητής: elastic packrat

Συμπεράσματα

- Αδιαφιλονίκητος νικητής: elastic packrat
- Καλύτερος χρόνος στα μεγάλα προγράμματα + σταθερή μνήμη

Συμπεράσματα

- Αδιαφιλονίκητος νικητής: elastic packrat
- Καλύτερος χρόνος στα μεγάλα προγράμματα + σταθερή μνήμη
- Στον παράλληλο δεν παρατηρείται επιτάχυνση (speedup) σε σχέση με τη σειριακή περίπτωση

Συμπεράσματα

- Αδιαφιλονίκητος νικητής: elastic packrat
- Καλύτερος χρόνος στα μεγάλα προγράμματα + σταθερή μνήμη
- Στον παράλληλο δεν παρατηρείται επιτάχυνση (speedup) σε σχέση με τη σειριακή περίπτωση
- Η πράξη της διατεταγμένης επιλογής, παρόλο που θεωρητικά αφήνει χώρο για παράλληλη εκτέλεση, δεν συμπεριφέρεται τόσο καλά στην πράξη

Συμπεράσματα

- Αδιαφιλονίκητος νικητής: elastic packrat
- Καλύτερος χρόνος στα μεγάλα προγράμματα + σταθερή μνήμη
- Στον παράλληλο δεν παρατηρείται επιτάχυνση (speedup) σε σχέση με τη σειριακή περίπτωση
- Η πράξη της διατεταγμένης επιλογής, παρόλο που θεωρητικά αφήνει χώρο για παράλληλη εκτέλεση, δεν συμπεριφέρεται τόσο καλά στην πράξη
- Overhead από fork-join

Συμπεράσματα

- Αδιαφιλονίκητος νικητής: elastic packrat
- Καλύτερος χρόνος στα μεγάλα προγράμματα + σταθερή μνήμη
- Στον παράλληλο δεν παρατηρείται επιτάχυνση (speedup) σε σχέση με τη σειριακή περίπτωση
- Η πράξη της διατεταγμένης επιλογής, παρόλο που θεωρητικά αφήνει χώρο για παράλληλη εκτέλεση, δεν συμπεριφέρεται τόσο καλά στην πράξη
- Overhead από fork-join
- Ίσως πετυχαίνουν συχνά στη γραμματική μας οι πρώτες (αν όχι η πρώτη) επιλογή, οπότε μετά υπάρχει και ο επιπλέον χρόνος αναμονής νημάτων

Συμπεράσματα

- Αδιαφιλονίκητος νικητής: elastic packrat
- Καλύτερος χρόνος στα μεγάλα προγράμματα + σταθερή μνήμη
- Στον παράλληλο δεν παρατηρείται επιτάχυνση (speedup) σε σχέση με τη σειριακή περίπτωση
- Η πράξη της διατεταγμένης επιλογής, παρόλο που θεωρητικά αφήνει χώρο για παράλληλη εκτέλεση, δεν συμπεριφέρεται τόσο καλά στην πράξη
- Overhead από fork-join
- Ίσως πετυχαίνουν συχνά στη γραμματική μας οι πρώτες (αν όχι η πρώτη) επιλογή, οπότε μετά υπάρχει και ο επιπλέον χρόνος αναμονής νημάτων
- Ο περιοδικό έλεγχος για τερματισμό αυξάνει το χρόνο εκτέλεσης

Απορίες

Απορίες

Ευχαριστώ για την προσοχή σας![1]

[1] Hope you slept well!