Τμήμα Μηχανικών Η/Υ & Πληροφορικής

Πανεπιστήμιο Πατρών

Παράλληλη Επεξεργασία

Ακαδημαϊκό Έτος 2019-2020

Νταβρούρος Αλέξιος ΑΜ: 1059653

Δημητρούκα Γιαννούλα ΑΜ: 1043770

Κωνσταντίνος Λιόπετας ΑΜ: 236113

Δομή Προγράμματος

Αφού διαβαστεί το αρχείο, όλοι οι κόμβοι αποθηκεύονται σε έναν πίνακα υπό τη μορφή μιας δομής.

```
10
11  typedef struct {
12    int num;
13    int va8mos;
14    int s;  //  size of geitones (int)
15    int b_size;  //  se bytes,gia geitones
16    int *geitones;  //  ari8mos geitona
17    omp_lock_t nlock;
18  }node;
19
```

```
//done
void stk_init(node *arr,int num,m_stack *stk){
  //stk->on_process = 0;
  stk->num = 0;
  stk->b_size = 1;
  stk->stk = (node *)malloc(sizeof(node *));
  for(int i = 0; i<num; i++){
    if(arr[i].va8mos == 0){

      stk->b_size+=sizeof(arr[i]);
      stk->stk=(node *)realloc(stk->stk,stk->b_size);
      stk->stk[stk->num] = arr[i];
      stk->num+=1;

    }
    else{continue;}

  }
}
```

Όσοι κόμβοι έχουν βαθμό εισόδου 0, μπαίνουν σε μια κοινή για όλα τα νήματα στοίβα.

Ένα μόνο νήμα δημιουργεί tasks, τα οποία περιέχουν ένα pop από την κοινή για κάθε νήμα στοίβα, πριν εκτελεστεί ο αλγόριθμος.

```
      #pragma omp taskwait
      {
        m_stack temp_stack;
        node on_process;

        omp_set_lock(&main_stack.nlock);
        on_process = stk_pop(&main_stack);
        omp_unset_lock(&main_stack.nlock);

        temp_stack = kahn(arr,&on_process);
```

Επειδή η στοίβα είναι κοινή για κάθε νήμα, γίνεται χρήση locks. Επίσης για τη στοίβα υπάρχει Rare Condition.

Κάθε νήμα έχει μια δική του στοίβα στην οποία αποθηκεύει τους γείτονες του κόμβου που βρίσκεται σε επεξεργασία.

```
  m_stack stack;
  //arxikopoihsh ths stoivas
  stack.num = 0;
  stack.b_size=0;
  stack.stk = (node *)malloc(sizeof(node *));

  if(sorted->num == -1){return stack;}
  // Main loop
  for(int i = sorted->s; i>0; i--){
    //vlepw poioi komvoi exoun va8mo eisodou 0
    omp_set_lock(&graph[ sorted->geitones[sorted->s-i] ].nlock);
    graph[ sorted->geitones[sorted->s-i] ].va8mos -= 1;
    //sto shmeio auto uparxei mono 1 thread to
    //opoio vlepei pws o va8mos eisodou einai isos me 0
    if(graph[ sorted->geitones[sorted->s-i] ].va8mos == 0){
      node temp = graph[ sorted->geitones[sorted->s - i] ];
      //printf("O kahn vrike : %i\n",temp.num );
      stk_push(&stack ,&temp );
    }
    omp_unset_lock(&graph[ sorted->geitones[sorted->s-i] ].nlock);

  }
  //printf("Size of stack : %i\n", stack.num);

  return stack;
}
```

Υπάρχει ένα lock για κάθε κόμβο του γραφήματος και όλοι οι κόμβοι αποθηκεύονται σε κοινή περιοχή. Όταν ένα νήμα επεξεργάζεται τον βαθμό εισόδου ενός κόμβου αυτό πρέπει να γίνεται σειριακά. Οι κόμβοι με μη κοινούς γείτονες επεξεργάζονται πιο γρήγορα απ' ότι κόμβοι με κοινούς γείτονες.

Μετά την επεξεργασία, το νήμα έχει στην ιδιωτική του στοίβα κόμβους, οι οποίοι έχουν βαθμό εισόδου 0, τους οποίους τοποθετεί στην κοινή για όλα τα νήματα στοίβα. Τότε θα γίνουν και push στην κοινή στοίβα.

```
if( temp_stack.num!=0 ){
  limit_neighbours(&arr[on_process.num-1] ,&temp_stack);
  omp_set_lock(&main_stack.nlock);
  while(temp_stack.num>0){
    node temp = stk_pop(&temp_stack);
    //printf("8a mpei se main %i\n", temp.num);
    stk_push(&main_stack,&temp);
  }
  omp_unset_lock(&main_stack.nlock);
}
```

Κάθε νήμα καλεί μια συνάρτηση, η οποία αφαιρεί από τον κόμβο που επεξεργάστηκε τους γείτονες που δεν έχουν βαθμό εισόδου 0, διότι δεν ανακαλύφθηκαν λόγω αυτού του κόμβου.

```
void limit_neighbours(node *komvos,m_stack *stk){
    //arxikopoihsh komvou
    komvos->geitones = (int *)malloc(stk->b_size);
    komvos->b_size = stk->b_size;
    komvos->s = stk->num;
    //kratw tous geitones me va8mo eisodou = 0
    for(int i = 0; i<stk->num; i++){komvos->geitones[i] = stk->stk[i].num-1;}

    komvos->geitones = (int *)realloc(komvos->geitones,komvos->b_size + sizeof(int) );
    komvos->geitones[komvos->s] = -1;//to telos twn geitwnwn

}
```

Μόνο ένα νήμα, το κύριο, εμφανίζει μέσω συναρτήσεων τους κόμβους με τη σειρά που ανακαλύφθηκαν.

```
#pragma omp master
{
  end = clock();
  double cpu_time_used = ((double) (end - start)) / CLOCKS_PER_SEC;
  while(arxikoi.num>0){
    node temp = stk_pop(&arxikoi);
    temp = arr[temp.num-1];
    print_a_node(arr,&temp);
  }
  printf("Xronos : %f sec.\n", cpu_time_used);
  exit(0);

}
```

```
void print_a_node(node *graph,node *n){
  printf("Node Number : %i\n",n->num );
  printf("Neigbour Number : %i\n",n->s );
  for(int i = 0; i<n->s; i++){
    node temp = graph[n->geitones[i]];
    print_a_node(graph,&temp );
  }
}
```

Στο παράλληλο αλλά και στο σειριακό πρόγραμμα συνολικά συμβαίνουν (αν και n είναι το πλήθος των κόμβων) n push και N pop σειριακά.

Χρόνος εκτέλεσης O(2N)=O(N).

Η επεξεργασία κάθε κόμβου εξαρτάται από το πλήθος των γειτόνων κάθε κόμβου που εξετάζεται. Έστω {Kn} ακολουθία με πλήθος_γειτόνων=Ki, i ανήκει σε {1, 2, 3,…,n}, n=Πλήθος_κόμβων. Άρα ένα νήμα επεξεργάζεται κάθε κόμβο i για διάστημα Ki. Αν υπάρχουν m κόμβοι μέσα σε κοινή για όλα τα νήματα στοίβα, τότε ο χρόνος επεξεργασίας όλων των κόμβων σε αυτή τη στοίβα είναι ίσος με B/A.

B= ΣKi, i ανήκει σε {1, 2, 3,…,n}, n=Πλήθος_κόμβων.

A= πλήθος νημάτων.

Για μία δεδομένη στιγμή που έχω στη στοίβα m κόμβους.

$$Speedup = \frac{1}{s + \frac{1 - s}{p}}$$

S=2m (push και pop) + (η πιθανότητα να έχουν δύο κόμβοι μέσα στη στοίβα κοινούς γείτονες)

(1-s)=(Οι διαφορετικοί γείτονες των κόμβων, που βρίσκονται σε επεξεργασία)

P= πλήθος νημάτων

Γραφήματα

Για το γράφημα **wb-cs-stanford.mtx:**

Πλήθος κόμβων: 9914

Κόμβοι που επεξεργάστηκαν: 892

Χωρίς Βελτιστοποίηση (-O0)

1	2	4	6	8
0.000131	0.000299	0.000223	0.000284	0.000320
0.000216	0.000167	0.000173	0.000291	0.000151
0.000212	0.000399	0.000097	0.000290	0.000174
0.000182	0.000185	0.000175	0.000289	0.000283
0.000209	0.000306	0.000216	0.000145	0.000384
0.000314	0.000428	0.000081	0.000316	0.000380
0.000301	0.000217	0.000193	0.000170	0.000181
0.000248	0.000304	0.000177	0.000322	0.000311

0.000170	0.000281	0.000232	0.000170	0.000180
0.000346	0.000286	0.000090	0.000179	0.000185
0.0002329	**0.0002872**	**0.0001657**	**0.0002456**	**0.0002549**

Με Βελτιστοποιήσεις (-O3)

1	2	4	6	8
0.000165	0.000056	0.000219	0.000183	0.000189
0.000273	0.000082	0.000125	0.000188	0.000216
0.000172	0.000153	0.000137	0.000107	0.000209
0.000255	0.000231	0.000184	0.000173	0.000191
0.000181	0.000185	0.000233	0.000130	0.000153
0.000268	0.000141	0.000142	0.000041	0.000277
0.000168	0.000182	0.000179	0.000012	0.000235
0.000173	0.000062	0.000231	0.000303	0.000313
0.000260	0.000140	0.000232	0.000185	0.000192
0.000233	0.000177	0.000168	0.000187	0.000243
0.0002148	**0.0001409**	**0.0001850**	**0.0001509**	**0.0002218**

Για το γράφημα **Roget.mtx**:

Πλήθος κόμβων: 1022

Κόμβοι που επεξεργάστηκαν: 152

Χωρίς Βελτιστοποίηση (-O0)

1	2	4	6	8
0.000043	0.000075	0.000051	0.000087	0.000044
0.000048	0.000043	0.000004	0.000068	0.000112
0.000077	0.000042	0.000043	0.000015	0.000101
0.000037	0.000037	0.000103	0.000045	0.000073
0.000005	0.000018	0.000036	0.000043	0.000033
0.000067	0.000074	0.000088	0.000071	0.000092
0.000082	0.000085	0.000097	0.000079	0.000089
0.000076	0.000048	0.000050	0.000010	0.000027
0.000035	0.000081	0.000094	0.000054	0.000082
0.000024	0.000093	0.000010	0.000036	0.000047
0.0000494	**0.0000596**	**0.0000576**	**0.0000508**	**0.0000700**

Με Βελτιστοποιήσεις (-O3)

1	2	4	6	8
0.000064	0.000053	0.000055	0.000026	0.000111
0.000032	0.000040	0.000009	0.000095	0.000046

0.000094	0.000095	0.000042	0.000003	0.000050
0.000035	0.000043	0.000109	0.000044	0.000098
0.000087	0.000068	0.000040	0.000049	0.000069
0.000081	0.000075	0.000032	0.000080	0.000043
0.000095	0.000083	0.000073	0.000079	0.000068
0.000079	0.000086	0.000063	0.000021	0.000048
0.000082	0.000037	0.000113	0.000083	0.000078
0.000073	0.000054	0.000109	0.000041	0.000010
0.0000722	**0.0000634**	**0.0000645**	**0.00000521**	**0.00000621**

Για το γράφημα **Csphd.mtx**:

Πλήθος κόμβων: 1882

<u>Με Βελτιστοποιήσεις -O3</u>

1	2	4	6	8
0.000438	0.000410	0.000246	0.000257	0.000219
0.000220	0.000235	0.000204	0.000226	0.000267
0.000271	0.000212	0.000263	0.000290	0.000235
0.000203	0.000196	0.000240	0.000315	0.000302
0.000667	0.000245	0.000270	0.001018	0.000256

0.000209	0.000195	0.000321	0.000252	0.000221
0.000308	0.000172	0.000240	0.000211	0.000307
0.000434	0.000216	0.000212	0.000226	0.000230
0.000231	0.000224	0.000533	0.000297	0.000315
0.000216	0.000188	0.000243	0.000335	0.000197
0.0003197	**0.0002293**	**0.0002777**	**0.0003427**	**0.0002549**

Χωρίς Βελτιστοποιήσεις -O0

1	2	4	6	8
0.001130	0.000251	0.000374	0.000858	0.000275
0.000196	0.000199	0.000252	0.000281	0.000268
0.001289	0.000322	0.000257	0.001376	0.000496
0.001657	0.000223	0.001927	0.000244	0.000283
0.000149	0.001243	0.000209	0.000838	0.001291
0.000349	0.000224	0.000228	0.000263	0.000634
0.000204	0.001145	0.000233	0.000196	0.000257

0.000238	0.000882	0.000183	0.000294	0.000276
0.000732	0.000244	0.001029	0.001814	0.000310
0.000217	0.000296	0.000234	0.000290	0.000501
0.0006161	**0.0005029**	**0.0004926**	**0.0006454**	**0.0004591**

Για το γράφημα **GlossGT.mtx**:

Πλήθος κόμβων: 78

<u>Χωρίς Βελτιστοποιήσεις -O0</u>

1	2	4	6	8
0.000340	0.000337	0.000255	0.000281	0.000355
0.000300	0.000301	0.000220	0.000244	0.000266
0.000304	0.000227	0.000401	0.000319	0.000356
0.000223	0.000361	0.000346	0.000375	0.000348
0.000289	0.000315	0.000331	0.000231	0.000350
0.000307	0.000003	0.000347	0.000358	0.000350
0.000320	0.000233	0.000247	0.000383	0.000340

0.000314	0.000318	0.000252	0.000297	0.000255
0.000313	0.000345	0.000387	0.000316	0.000346
0.000214	0.000333	0.000325	0.000266	0.000242
0.0002924	**0.0002773**	**0.0003111**	**0.0003070**	**0.0003208**

Με Βελτιστοποιήσεις -O3

1	2	4	6	8
0.000233	0.000307	0.000232	0.000342	0.000255
0.000333	0.000301	0.000297	0.000261	0.000332
0.000312	0.000196	0.000321	0.000230	0.000246
0.000316	0.000354	0.000350	0.000228	0.000276
0.000205	0.000345	0.000353	0.000358	0.000384
0.000233	0.000258	0.000375	0.000347	0.000384
0.000220	0.000228	0.000322	0.000385	0.000355
0.000304	0.000364	0.000353	0.000325	0.000337

0.000496	0.000286	0.000373	0.000352	0.000366
0.000334	0.000330	0.000246	0.000321	0.000364
0.0002986	**0.0002969**	**0.0003222**	**0.0003149**	**0.0003299**

Για το γράφημα **GD95_a.mtx**:

<u>Χωρίς Βελτιστοποιήσεις -O0</u>

thread 1:

time=(0.001202+0.000012+0.000010+0.001691+0.001377 +0.001203 +0.000255 +0.000434+0.001239+0.001447)/10=0.000887 sec

threads 2:

time= (0.000951+0.001444+0.001507 +0.000231+0.001089+0.000430 +0.001463+0.001279+0.001197+0.001659)/10=0.001125 sec

threads 4:

time=(0.001089+0.000384+0.001488+ 0.001505+0.001411 + 0.000018+ 0.000591+0.001607 +0.001501+0.001501)/10=0.0011095 sec

threads 8:

time=(0.001381+ 0.000437+0.001606+0.001001+0.000319+0.000436+0.000504+ 0.000416+0.001047+ 0.000491)/10=0.0007638 sec

<u>Με Βελτιστοποιήσεις -O3</u>

thread 1:

time=(0.001261+0.001746 +0.000469+0.001400 +0.001602+0.000016 + 0.000585+0.000368 +0.000967+0.000350)/10=0.0008764 sec

threads 2:

time=(0.001443+ 0.000600+ 0.000425+0.000347 +0.000239 +0.001266
+0.000723+0.000354 +0.001354+0.000382)/10=0.0007133 sec

threads 4:

time=(0.000327 +0.000023 +0.000391+0.000449 +0.001516
+0.001452+0.001298+0.001318+ 0.001454+0.001686)/10=0.0009914 sec

threads 8:

time=(0.001426+0.001776+0.000470 +0.000464+0.001641+0.002076+0.001561+0.000329
+0.000461+ 0.000395)/10=0.0010599 sec

Threads	**1**	**2**	**4**	**8**
Time (seconds) *Χωρίς Βελτιστοποίηση (-O0)*	0.000887	0.001125	0.0011095	0.0007638
Time (seconds) *Με Βελτιστοποίηση (-O3)*	0.0008764	0.0007133	0.0009914	0.0010599

Για το γράφημα **mycielskian3.mtx:**

Χωρίς Βελτιστοποιήσεις -O0

thread 1:

time=(0.001027+0.000082 +0.000810+ 0.001399+0.000630 +0.000140
+0.000765+0.001105+0.000053+0.001221)/10=0.0007232 sec

threads 2:

time=(0.000116+0.000327+ 0.001290+0.002781+0.001723+0.001856+0.000183+0.000693+ 0.000085+0.001335)/10=0.0010389 sec

threads 4:

time=(0.000649 + 0.001689 + 0.001581+0.000084+ 0.000114+ 0.001234+0.000015+0.000023+0.001343+0.000011)/10=0.0007 sec

threads 8:

time=(0.000861+0.000244+ 0.000138+0.001001+ 0.000206+0.000011+0.001272+0.000083 +0.000091 + 0.001043+0.001721)/10=0.0006671 sec

Με Βελτιστοποιήσεις -O3

thread 1:

time=(0.001387+0.001078+ 0.000424+0.001189+0.000132+0.001647+0.000091 +0.000407

+0.002321 + 0.001598)/10=0.0010274 sec

threads 2:

time=(0.001281+0.000087+0.001741+0.000152+0.001397 +0.001467+0.000045+0.001373 +0.001690+ 0.001151)/10=0.0010384 sec

threads 4:

time=(0.000392 +0.000008+0.000019+ 0.001697+0.000013+0.000017+0.001183 +0.001266+0.001445+0.001239)/10=0.0007279 sec

threads 8:

time=(0.000216 + 0.001677+ 0.000198+0.000281+0.002472+0.000816+0.001415 +0.000023+0.000016 +0.000243)/10=0.0007357 sections

Threads	**1**	**2**	**4**	**8**
Time (seconds) *Χωρίς Βελτιστοποίηση (-O0)*	0.0007232	0.0010389	0.0007	0.0006671

Time (seconds) *Με Βελτιστοποίηση* *(-O3)*	0.0010274	0.0010384	0.0007279	0.0007357

Για το γράφημα **wiki-Vote.mtx** (κυκλικό)

Κόμβοι που επεξεργάστηκαν: 5981

Χωρίς Βελτιστοποιήσεις -O0

thread 1:

time=(0.000775+0.000648+0.000767 +0.000402+0.001287 + 0.001291+0.001091 +0.000489+ 0.000318 +0.001732)/10=0.00088 sec

threads 2:

time=(0.000447+ 0.000974+ 0.000952+0.000600+0.000477+0.000961+0.000263+0.000556+0.001049+0.001373)/10=0.0007652 sec

threads 4:

time=(0.000499+ 0.000847+ 0.001214+ 0.001126+ 0.001182+ 0.001601+ 0.000692+0.000766+0.001774+0.001701)/10=0.0011402 sec

threads 8:

time=(0.000497+ 0.001046+ 0.001160 +0.001799+0.001018+ 0.001311+ 0.001099+0.000915 +0.000130+0.001087)/10=0.0010062 sec

Με Βελτιστοποιήσεις -O3

thread 1:

time=(0.001124+0.000751+ 0.000784+0.001067+0.001695 +0.000962 +0.001403+0.000991 +0.000722 +0.000339)/10=0.0009838 sec

threads 2:

time=(0.000363+0.001235+ 0.000686+0.000682 +0.000309 +0.000716 + 0.000936+0.001190+

0.000689+ 0.000851)/10=0.0007657 sec

threads 4:

time=(0.000818 + 0.001208+ 0.000853 +0.000150+0.000889+0.001334 +0.001248+ 0.000706+ 0.000935+0.000664)/10=0.0008805 sec

threads 8:

time=(0.000798 +0.000685+0.001000+ 0.000821+ 0.001336+0.000274+ 0.000285+0.000719+0.001359 +0.000670)/10=0.0007947 sec

Threads	**1**	**2**	**4**	**8**
Time (seconds) *Χωρίς Βελτιστοποίηση (-O0)*	0.00088	0.0007652	0.0011402	0.0010062
Time (seconds) *Με Βελτιστοποίηση (-O3)*	0.0009838	0.0007657	0.0008805	0.0007947