CMPUT 481 Assignment 1

Brock Toews

October 1, 2013

1284088

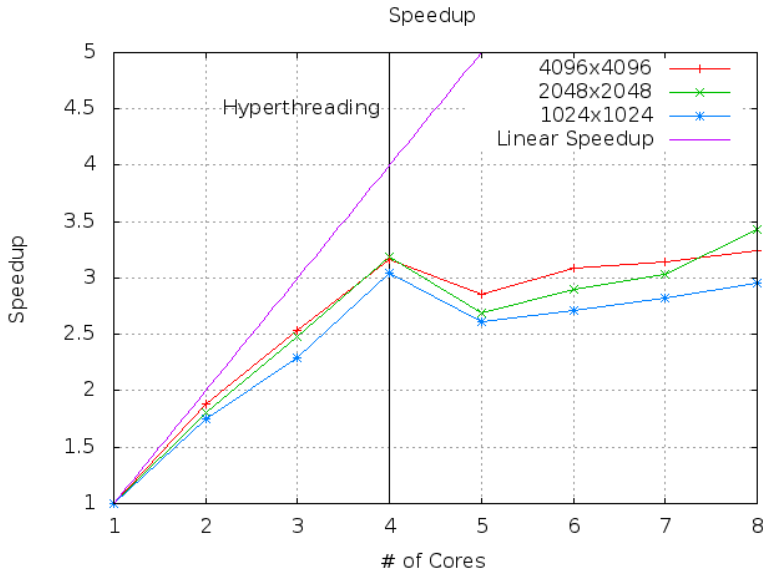

Figure 1: Chart of speedups

1 Introduction

2 Implementation

3 Implications

4 Charts

5 Source Code

5.1 Parallel Implementation

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <pthread.h>
4 #include <err.h>
5 #include <sys/time.h>
6
7 static void * gen_seg(void *);
8 static void * mult_seg(void *);
```

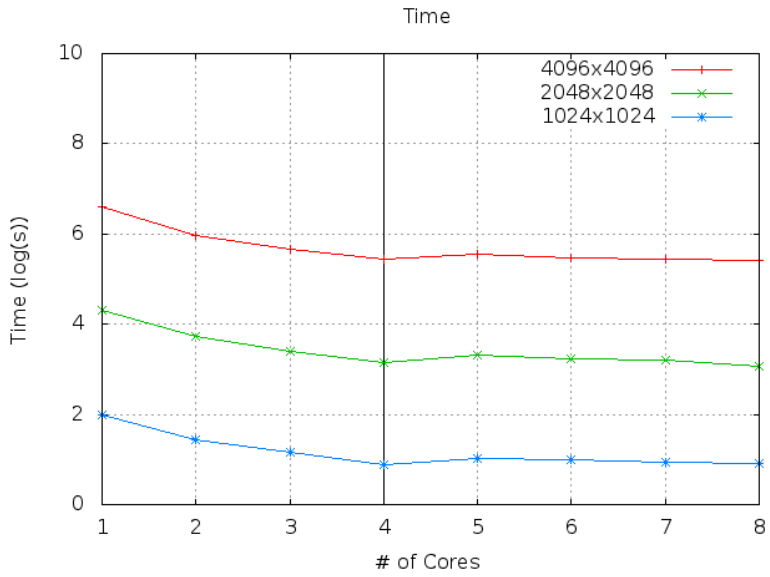

Figure 2: Chart of times to multiply matrices

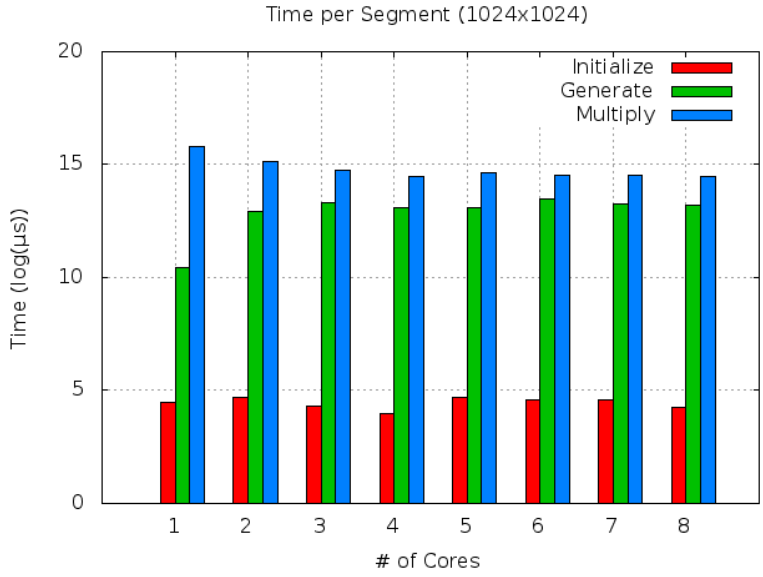

Figure 3: Chart of times to complete segments

```
 9
10 static pthread_barrier_t bar_gen;
11 static pthread_barrier_t bar_mult;
12 static int nproc;
13 static int mat_size;
14 static long * mat1;
15 static long * mat2;
16 static long * mat_fin;
17
18 struct seg_desc {
19         int start;
20         int end;
21 };
22
23 int main(int argc, char * argv[]) {
24         struct seg_desc * segs;
25         struct timeval start_fin;
26         struct timeval end_fin;
27         struct timeval init_fin;
28         struct timeval gen_fin;
29         int init_sec;
30         int init_usec;
31         double init_time;
32         int gen_sec;
33         int gen_usec;
34         double gen_time;
35         int mult_sec;
36         int mult_usec;
37         double mult_time;
38         int tot_sec;
39         int tot_usec;
40         double tot_time;
41         int i;
42         int seg_size;
43         pthread_t * thread_ids;
44
45         /*start timing*/
46         gettimeofday(&start_fin, NULL);
47
48         /*handle user input*/
```

```
49          if (argc == 2) {
50                  nproc = strtol(argv[1], NULL, 10);
51          } else if (argc == 3) {
52                  nproc = strtol(argv[1], NULL, 10);
53                  mat_size = strtol(argv[2], NULL, 10);
54          } else {
55                  nproc = get_nprocs();
56                  mat_size = 1024;
57          }
58
59          segs = malloc(sizeof(struct seg_desc)*nproc);
60          thread_ids = malloc(sizeof(pthread_t)*nproc);
61          mat1 = malloc(sizeof(long)*mat_size*mat_size);
62          mat2 = malloc(sizeof(long)*mat_size*mat_size);
63          mat_fin = malloc(sizeof(long)*mat_size*mat_size);
64
65          pthread_setconcurrency(nproc);
66
67          /*Choose row assignments*/
68          seg_size = mat_size/nproc;
69          for (i = 0; i < nproc; i ++) {
70                  segs[i].start = i*seg_size;
71                  segs[i].end = (i + 1)*seg_size;
72                  if (i == nproc - 1) {
73                          segs[i].end += mat_size % nproc;
74                  }
75          }
76
77          gettimeofday(&init_fin, NULL);
78
79          /*Generate matrices*/
80          for (i = 0; i < nproc; i ++) {
81                  pthread_create(&thread_ids[i], NULL, &gen_seg, &segs[i]);
82          }
83
84          for (i = 0; i < nproc; i ++) {
85                  pthread_join(thread_ids[i], NULL);
86          }
87
88          gettimeofday(&gen_fin, NULL);
```

```
89
90          /*Multiply Matrices*/
91          for (i = 0; i < nproc; i ++) {
92                  pthread_create(&thread_ids[i], NULL, &mult_seg, &segs[i]);
93          }
94
95          for (i = 0; i < nproc; i ++) {
96                  pthread_join(thread_ids[i], NULL);
97          }
98
99          /*end timing*/
100         gettimeofday(&end_fin, NULL);
101
102         tot_sec = (int)end_fin.tv_sec - (int)start_fin.tv_sec;
103         tot_usec = (int)end_fin.tv_usec - (int)start_fin.tv_usec;
104         tot_time = (double)tot_sec + ((double)tot_usec/1000000.0);
105
106         init_sec = (int)init_fin.tv_sec - (int)start_fin.tv_sec;
107         init_usec = (int)init_fin.tv_usec - (int)start_fin.tv_usec;
108         init_time = (double)init_sec + ((double)init_usec/1000000.0);
109
110         gen_sec = (int)gen_fin.tv_sec - (int)init_fin.tv_sec;
111         gen_usec = (int)gen_fin.tv_usec - (int)init_fin.tv_usec;
112         gen_time = (double)gen_sec + ((double)gen_usec/1000000.0);
113
114         mult_sec = (int)end_fin.tv_sec - (int)gen_fin.tv_sec;
115         mult_usec = (int)end_fin.tv_usec - (int)gen_fin.tv_usec;
116         mult_time = (double)mult_sec + ((double)mult_usec/1000000.0);
117
118         printf("%lf %lf %lf %lf %d %d\n", tot_time, init_time, gen_time,
119         mult_time, nproc, mat_size);
120
121         return 0;
122 }
123
124 /*generates a given segment of a matrix*/
125 static void * gen_seg(void * arg) {
126         int row;
127         int col;
128         struct seg_desc seg = *((struct seg_desc*)arg);
```

```
129
130         srandom(time(0));
131
132         for (row = seg.start; row < seg.end; row ++) {
133                 for (col = 0; col < mat_size; col ++) {
134                         mat1[col + row*mat_size] = random();
135                         mat2[col + row*mat_size] = random();
136                 }
137         }
138
139         pthread_exit(0);
140 }
141
142 /*multiples a given segment of mat1 to the corresponding segment of mat2
143  *and places the result in mat_fin
144  */
145 static void * mult_seg(void * arg) {
146         int row;
147         int col;
148         int cell;
149         struct seg_desc seg = *((struct seg_desc*)arg);
150
151         for (row = seg.start; row < seg.end; row ++) {
152                 for (col = 0; col < mat_size; col ++) {
153                         int res_ind = row*mat_size + col;
154                         mat_fin[res_ind] = 0;
155                         for (cell = 0; cell < mat_size; cell ++) {
156                                 int ind1 = row*mat_size + cell;
157                                 int ind2 = cell*mat_size + col;
158                                 long prod = mat1[ind1] * mat2[ind2];
159                                 mat_fin[res_ind] += prod;
160                         }
161                 }
162         }
163
164         pthread_exit(0);
165 }
```

5.2 Sequential Implementation

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <err.h>
4  #include <sys/time.h>
5
6  int main(int argc, char * argv[]) {
7          int row;
8          int col;
9          int cell;
10         int mat_size;
11         long * mat1;
12         long * mat2;
13         long * mat_fin;
14         struct timeval start_fin;
15         struct timeval init_fin;
16         struct timeval gen_fin;
17         struct timeval end_fin;
18         int init_sec;
19         int init_usec;
20         double init_time;
21         int gen_sec;
22         int gen_usec;
23         double gen_time;
24         int mult_sec;
25         int mult_usec;
26         double mult_time;
27         int tot_sec;
28         int tot_usec;
29         double tot_time;
30
31         /*start timing*/
32         gettimeofday(&start_fin, NULL);
33
34         if (argc == 2) {
35                 mat_size = strtol(argv[1], NULL, 10);
36         } else {
37                 mat_size = 1024;
38         }
```

```
39
40          mat1 = malloc(sizeof(long)*mat_size*mat_size);
41          mat2 = malloc(sizeof(long)*mat_size*mat_size);
42          mat_fin = malloc(sizeof(long)*mat_size*mat_size);
43
44          /*Generate matrices*/
45          srandom(time(0));
46
47          gettimeofday(&init_fin, NULL);
48
49          for (row = 0; row < mat_size; row ++) {
50                  for (col = 0; col < mat_size; col ++) {
51                          mat1[col + row*mat_size] = random();
52                          mat2[col + row*mat_size] = random();
53                  }
54          }
55
56          gettimeofday(&gen_fin, NULL);
57
58          /*Multiply Matrices*/
59          for (row = 0; row < mat_size; row ++) {
60                  for (col = 0; col < mat_size; col ++) {
61                          int res_ind = row*mat_size + col;
62                          mat_fin[res_ind] = 0;
63                          for (cell = 0; cell < mat_size; cell ++) {
64                                  int ind1 = row*mat_size + cell;
65                                  int ind2 = cell*mat_size + col;
66                                  long prod = mat1[ind1] * mat2[ind2];
67                                  mat_fin[res_ind] += prod;
68                          }
69                  }
70          }
71
72          /*end timing*/
73          gettimeofday(&end_fin, NULL);
74
75          tot_sec = end_fin.tv_sec - start_fin.tv_sec;
76          tot_usec = end_fin.tv_usec - start_fin.tv_usec;
77          tot_time = (double)tot_sec + ((double)tot_usec/1000000.0);
78
```

```
79          init_sec = init_fin.tv_sec - start_fin.tv_sec;
80          init_usec = init_fin.tv_usec - start_fin.tv_usec;
81          init_time = (double)init_sec + ((double)init_usec/1000000.0);
82
83          gen_sec = gen_fin.tv_sec - init_fin.tv_sec;
84          gen_usec = gen_fin.tv_usec - init_fin.tv_usec;
85          gen_time = (double)gen_sec + ((double)gen_usec/1000000.0);
86
87          mult_sec = end_fin.tv_sec - gen_fin.tv_sec;
88          mult_usec = end_fin.tv_usec - gen_fin.tv_usec;
89          mult_time = (double)mult_sec + ((double)mult_usec/1000000.0);
90
91          printf("%lf %lf %lf %lf %d %d\n", tot_time, init_time, gen_time,
92          mult_time, 1, mat_size);
93
94          return 0;
95  }
```