Rohan Jhaveri

830962238

Assignment 3

Exercise 1:

Q1)

Yes when a C code is given as an input to the algorithm it will always produce a schedule for each statement. When a C code I given as an input, the algorithm first builds the legality constraints for each statements using Farkas Lemma technique. Using these constraints we build a communication model and calculate the reuse distance. Once we find the constraints for all the statement we apply the lexicographic minimal algorithm to find the minimum solution to find all the independent schedules. Hence we can find schedules for all the statements in the C code.

Q2)

a) Yes, the first loop dimension models the entire loop structure. In a nested loop structure, the first loop dimension is interpreted as hours and is the most important dimension. The next loop dimension is interpreted as minutes and so on. Hence first loop dimension models the entire loop structure.

b) In order to have legal schedule, the first loop dimensions should always happen before the second loop. Hence, it should be strongly satisfy the dependence condition or it should be parallel as in this case where the first loop dimension is zero. So in this case it does not matter when is the first loop dimension executed and hence there is no strongly satisfying dependence condition.

Rohan Jhaveri

830962238

Assignment 3

Exercise 2:

gcc -O3 polybench-c-4.2.1-beta/linear-algebra/blas/gemm/gemm.c polybench-c-4.2.1-beta/utilities/polybench.c -Ipolybench-c-4.2.1-beta/utilities -DPOLYBENCH_TIME -o gemm.orig

./gemm.orig

Execution time for gemm.orig: 0.793554

./bin/pocc --verbose polybench-c-4.2.1-beta/linear-algebra/blas/gemm/gemm.c

gcc -O3 polybench-c-4.2.1-beta/linear-algebra/blas/gemm.pocc.c polybench-c-4.2.1-beta/utilities/polybench.c -Ipolybench-c-4.2.1-beta/utilities -DPOLYBENCH_TIME -o gemm.transfo

./gemm.transfo

Execution time for gemm.transfo: 0.790434

Q1)

a) ./bin/pocc --verbose --pluto --pluto-fuse maxfuse polybench-c-4.2.1-beta/linear-algebra/blas/gemm/gemm.c

gcc -O3 polybench-c-4.2.1-beta/linear-algebra/blas/gemm.pocc.c polybench-c-4.2.1-beta/utilities/polybench.c -Ipolybench-c-4.2.1-beta/utilities -DPOLYBENCH_TIME -o gemm.transfo

./gemm.transfo

Execution time for gemm.transfo: 12.488275

b) As we, observed in (a) the time require increases to 12.488275 which is much slower that than the original code which is 0.793554. This because, sometimes the fusion can increase the pressure on registers when large loops are fused together. This specially happens in 3D loop structures.

c) The schedule found by pluto shows that dimension c0, dimension c1 and dimension c4 are parallel.

d) ./bin/pocc --verbose --pluto --pluto-fuse smartfuse polybench-c-4.2.1-beta/linear-algebra/blas/gemm/gemm.c

gcc -O3 polybench-c-4.2.1-beta/linear-algebra/blas/gemm.pocc.c polybench-c-4.2.1-beta/utilities/polybench.c -Ipolybench-c-4.2.1-beta/utilities -DPOLYBENCH_TIME -o gemm.transfo

./gemm.transfo

Execution Time for gemm.transfo: 12.452966

In this case, the execution time is slightly better than the (a) maxfuse as the loop fusion is done in a smarter way than maxfuse in which it fuses all the loops which are possible, but still the performance is bad as compared to the original code without any fusion.

Rohan Jhaveri

830962238

Assignment 3

Q2)

a) Default tile size

```
./bin/pocc --verbose --pluto --pluto-fuse maxfuse --pluto-tile polybench-c-4.2.1-beta/linear-algebra/blas/gemm/gemm.c

gcc -O3 polybench-c-4.2.1-beta/linear-algebra/blas/gemm.pocc.c polybench-c-4.2.1-beta/utilities/polybench.c -Ipolybench-c-4.2.1-beta/utilities -DPOLYBENCH_TIME -o gemm.transfo

./gemm.transfo
```

Execution time for gemm.transfo: 1.711478

b) Yes, in this case the performance is better than smartfuse as we do tiling along with fusion. So the locality and data reuse increases and hence the performance is improved as compared to stand alone smartfuse or maxfuse flags.

Q3)

a) For tile size 50 80 64

```
./bin/pocc --verbose --pluto --pluto-fuse maxfuse --pluto-tile tile polybench-c-4.2.1-beta/linear-algebra/blas/gemm/gemm.c

gcc -O3 polybench-c-4.2.1-beta/linear-algebra/blas/gemm.pocc.c polybench-c-4.2.1-beta/utilities/polybench.c -Ipolybench-c-4.2.1-beta/utilities -DPOLYBENCH_TIME -o gemm.transfo

./gemm.transfo
```

Execution time for gemm.transfo: 1.921862

In this case when we apply the tile of size 50 80 64, the performance is better than smartfuse or maxfuse alone but it degrades as compared to default tile size. This is due to the fact that data locality and reuse is decreased as compared to the default tile size.

b) For tile size 1 80 128

```
./bin/pocc --verbose --pluto --pluto-fuse maxfuse --pluto-tile tile polybench-c-4.2.1-beta/linear-algebra/blas/gemm/gemm.c

gcc -O3 polybench-c-4.2.1-beta/linear-algebra/blas/gemm.pocc.c polybench-c-4.2.1-beta/utilities/polybench.c -Ipolybench-c-4.2.1-beta/utilities -DPOLYBENCH_TIME -o gemm.transfo

./gemm.transfo
```

Execution time for gemm.transfo: 2.607909

In this case the performance id degraded even further as compared to 1 80 128. This is due to the fact that data locality and reuse is decreased as compared to the previous tile size.

Rohan Jhaveri

830962238

Assignment 3

Q4)

Script:

```
#! /usr/bin/python2
import numpy as np
import os
import sys
import subprocess
tile_size = ["1 80 128", "50 80 64", "4 4 1", "8 8 1", "4 4 8", "4 2 4", "4 2
2", "4 8 16", "8 4 16", "8 4 8", "4 2 8", "4 4 2", "4 4 4", "32 64 8", "32 64
16", "16 32 8", "128 224 32", "192 160 32", "48 72 12", "64 96 16"]
command1 = "./pocc-1.4/bin/pocc --verbose --pluto --pluto-fuse maxfuse --pluto-
tile  polybench-c-4.2.1-beta/linear-algebra/blas/gemm/gemm.c"
command2        =        "gcc        -O3        polybench-c-4.2.1-beta/linear-
algebra/blas/gemm/gemm.pocc.c  polybench-c-4.2.1-beta/utilities/polybench.c  -
Ipolybench-c-4.2.1-beta/utilities -DPOLYBENCH_TIME -o gemm.transfo"
command3 = "./gemm.transfo | awk '{print $1}'"
if(os.path.isfile('opfile.txt')):
    os.remove('opfile.txt')
for i in range(len(tile_size)):
    with open('tile.sizes', 'w') as inp:
        inp.write(str(tile_size[i]))
    print tile_size[i]
    os.system(command1)
    os.system(command2)
    with open('opfile.txt','a') as out:
            x = subprocess.check_output(command3,shell = True)
            out.write(str(tile_size[i])+ '->'+x)
d = {}
with open ('opfile.txt', 'r') as op:
    for line in op:
        (key,val) = line.split('->')
        d[key] = float(val.replace('\n', ''))
a =  sorted(d.iteritems(),key = lambda (k,v):(v,k))
for key,val in a:
        print "%s: %f"% (key,val)
print a[0]
```

Rohan Jhaveri

830962238

Assignment 3

Q5)

20 different tile sizes explored are given in the list below:

Tile_size = ["1 80 128", "50 80 64", "4 4 1", "8 8 1", "4 4 8", "4 2 4", "4 2 2", "4 8 16", "8 4 16", "8 4 8", "4 2 8", "4 4 2", "4 4 4", "32 64 8", "32 64 16", "16 32 8", "128 224 32", "192 160 32", "48 72 12", "64 96 16"]

Best performance = Tile size: 48 72 12, Execution time: 1.627871

Reason: The main purpose of tiling is to increase the data locality and data reuse. In order to achieve that we need to do tiling such that the all the data is in inside L1 and L2 cache. Tile size ae determined by the (items read by work group x items read during first iteration of internal loop for a work group x items read during first iteration of internal loop of work item) along dimension 0 of the loop structure. This lead to increase in data localization and reuse.

So, I started working with dimensions provided and then as I understood the pattern I imagined how a cache would look like in a dgemm kernel and I started exploring these tile sizes as given in the list.

Rohan Jhaveri

830962238

Assignment 3

Exercise 3:
Q1)
1) Script:

```
#! /usr/bin/python2
import numpy as np
import os
import sys
import subprocess
tile_size = ["1 80 128", "50 80 64", "4 4 1", "8 8 1", "4 4 8", "192 32 15",
"15 192 32", "480 480 480", "16 16 256", "3 3 2", "3 3 480", "3 3 2", "3 2
480", "4 4 2", "16 32 256", "1 2 4", "1 2 480", "1 4 480", "2 4 480", "4 4 1"]
commands = ['./pocc-1.4/bin/pocc --verbose --pluto-fuse maxfuse --pluto-tile
polybench-c-4.2.1-beta/stencils/heat-3d/heat-3d.c',
'./pocc-1.4/bin/pocc --verbose --pluto-fuse maxfuse --pluto-parallel --pluto-
tile polybench-c-4.2.1-beta/stencils/heat-3d/heat-3d.c',
'./pocc-1.4/bin/pocc --verbose --pluto-fuse maxfuse --pragmatizer --pluto-tile
polybench-c-4.2.1-beta/stencils/heat-3d/heat-3d.c',
'./pocc-1.4/bin/pocc --verbose --pluto-fuse maxfuse --pragmatizer --pluto-
parallel --pluto-tile polybench-c-4.2.1-beta/stencils/heat-3d/heat-3d.c',
'./pocc-1.4/bin/pocc --verbose --pluto-fuse smartfuse --pluto-tile polybench-
c-4.2.1-beta/stencils/heat-3d/heat-3d.c',
'./pocc-1.4/bin/pocc --verbose --pluto-fuse smartfuse --pluto-parallel --
pluto-tile polybench-c-4.2.1-beta/stencils/heat-3d/heat-3d.c',
'./pocc-1.4/bin/pocc --verbose --pluto-fuse smartfuse --pragmatizer --pluto-
tile polybench-c-4.2.1-beta/stencils/heat-3d/heat-3d.c',
'./pocc-1.4/bin/pocc --verbose --pluto-fuse smartfuse --pragmatizer --pluto-
parallel --pluto-tile polybench-c-4.2.1-beta/stencils/heat-3d/heat-3d.c',
'./pocc-1.4/bin/pocc --verbose --pluto-fuse nofuse --pluto-tile polybench-c-
4.2.1-beta/stencils/heat-3d/heat-3d.c',
'./pocc-1.4/bin/pocc --verbose --pluto-fuse nofuse --pluto-parallel --pluto-
tile polybench-c-4.2.1-beta/stencils/heat-3d/heat-3d.c',
'./pocc-1.4/bin/pocc --verbose --pluto-fuse nofuse --pragmatizer --pluto-tile
polybench-c-4.2.1-beta/stencils/heat-3d/heat-3d.c',
'./pocc-1.4/bin/pocc --verbose --pluto-fuse nofuse --pragmatizer --pluto-
parallel --pluto-tile polybench-c-4.2.1-beta/stencils/heat-3d/heat-3d.c']
command2 = "gcc -O3 polybench-c-4.2.1-beta/stencils/heat-3d/heat-3d.pocc.c
polybench-c-4.2.1-beta/utilities/polybench.c             -Ipolybench-c-4.2.1-
beta/utilities -DPOLYBENCH_TIME -o heat-3d.transfo"
command4 = "gcc -O3 -fopenmp polybench-c-4.2.1-beta/stencils/heat-3d/heat-3d.c
polybench-c-4.2.1-beta/utilities/polybench.c             -Ipolybench-c-4.2.1-
beta/utilities -DPOLYBENCH_TIME -o heat-3d.par.transfo"
command3 = "./heat-3d.transfo | awk '{print $1}'"
command_list = ['./pocc-1.4/bin/pocc --verbose --pluto-fuse maxfuse --
pragmatizer --pluto-tile polybench-c-4.2.1-beta/stencils/heat-3d/heat-3d.c',
'./pocc-1.4/bin/pocc --verbose --pluto-fuse maxfuse --pragmatizer --pluto-
parallel --pluto-tile polybench-c-4.2.1-beta/stencils/heat-3d/heat-3d.c',
```

Rohan Jhaveri

830962238

Assignment 3

```
'./pocc-1.4/bin/pocc --verbose --pluto-fuse smartfuse --pragmatizer --pluto-
tile polybench-c-4.2.1-beta/stencils/heat-3d/heat-3d.c',
'./pocc-1.4/bin/pocc --verbose --pluto-fuse smartfuse --pragmatizer --pluto-
parallel --pluto-tile polybench-c-4.2.1-beta/stencils/heat-3d/heat-3d.c',
'./pocc-1.4/bin/pocc --verbose --pluto-fuse nofuse --pragmatizer --pluto-tile
polybench-c-4.2.1-beta/stencils/heat-3d/heat-3d.c',
'./pocc-1.4/bin/pocc  --verbose  --pluto-fuse  nofuse  --pragmatizer  --pluto-
parallel --pluto-tile polybench-c-4.2.1-beta/stencils/heat-3d/heat-3d.c']

if(os.path.isfile('opfile.txt')):
    os.remove('opfile.txt')
for x in commands:
        for i in range(len(tile_size)):
                with open('tile.sizes', 'w') as inp:
                        inp.write(str(tile_size[i]))
        print tile_size[i]
                os.system(x)
                if x in command_list:
                        os.system(command4)
                else:
                        os.system(command2)
                with open('opfile.txt','a') as out:
                        x = subprocess.check_output(command3,shell = True)
                        out.write(str(tile_size[i])+ '->'+x)
d = {}
with open ('opfile.txt', 'r') as op:
    for line in op:
        (key,val) = line.split('->')
        d[key] = float(val.replace('\n', ''))
a =  sorted(d.iteritems(),key = lambda (k,v):(v,k))
for key,val in a:
        print "%s: %f"% (key,val)
print a[0]
```

Rohan Jhaveri

830962238

Assignment 3

Q2)

Tile_sizes = ["1 80 128", "50 80 64", "4 4 1", "8 8 1", "4 4 8", "192 32 15", "15 192 32", "480 480 480", "16 16 256", "3 3 2", "3 3 480", "3 3 2", "3 2 480", "4 4 2", "16 32 256", "1 2 4", "1 2 480", "1 4 480", "2 4 480", "400 625 30"]

Reason: Again the reason for selecting these tile sizes is the same i.e. for cache locality and data reuse. I started exploring with similar sizes as dgemm kernel and as I found the dimensions requiring high locality and cache requirements for tiling I started exploring different tile sizes as mentioned above in the list above.

Q3)

The execution time for all the transformations are shown in opfile.txt file after the execution of the script which is shown below:

1 80 128 -> 7.113177
50 80 64 -> 7.422241
4 4 1 -> 7.046286
8 8 1 -> 7.123678
4 4 8 -> 7.467923
192 32 15 -> 6.492376
15 192 32 -> 7.533267
480 480 480 -> 6.781236
16 16 256 -> 7.517639
3 3 2 -> 7.3716562
3 3 480 -> 7.651861
3 3 1 -> 7.071619
3 2 480 -> 7.492787
4 4 2 -> 6.8614579
16 32 256 -> 7.473915
1 2 4 -> 7.811467
1 2 480 -> 7.417627
1 4 480 -> 7.327164
2 4 480 -> 7.548194
400 625 30 -> 6.113177

Q4)

Best Performance Tile_size = 400 625 30, Execution time = 6.113177.