CHALMERS
EXAMINATION / TENTAMEN

Course code/kurskod	Course name/kursnamn
DIT431	High Performance Parallel Programming

Anonymous code Anonym kod	Examination date Tentamensdatum	Number of pages Antal blad	Grade Betyg
DIT431-0007-ADT	27/10/2023	13	

I confirm that I've no mobile or other similar electronic equipment available during the examination.
Jag intygar att jag inte har mobiltelefon eller annan liknande elektronisk utrustning tillgänglig under eximinationen.

Solved task Behandlade uppgifter No/nr		Points per task Poäng på uppgiften	Observe: Areas with bold contour are to completed by the teacher. Anmärkning: Rutor inom bred kontur ifylles av lärare.
1	X	8	
2	X	8	
3	X	2	
4	X	9	
5	X	9	
6	X	6	
7			
8			
9			
10			
11			
12			
13			
14			
15			
16			
17			
Bonus poäng			
Total examination points Summa poäng på tentamen		42	

[redacted]

1

5

CHALMERS	Anonymous code / Anonym kod	Points for question (to be filled in by teacher) / Poäng på uppgiften (ifylles av lärare)	Consecutive page no. / Löpande sid nr
	DIT431-0007-ADT		1
			Question no. / Uppgift nr 1

~~(a) (1) N:1~~

~~1000 ULTs.~~

~~$\frac{1000}{10}$ = 100 ULTs on each core.~~

~~Total execution time = 100 * (10 + 1)~~

~~= 1100 units~~

(1) N:1:

(a) Only 1 core can be used.

1000 ULTs.

Total execution time = 1000 * (10 + 1)

= 11 000 units.

2.5

(2) 1:1

$\frac{1000}{10}$ = 100 ~~ULTs~~ KLTS on each core.

Total execution time = 100 (10 + 5)

= 1500 units.

(3) N:M, 10:1

No. of KLTs = $\frac{1000}{10}$ = 100.

~~total~~ execution time of 1 KLT = 10 * time for 1 ULT

= 10 * (10 + 1)

= 110 units

total execution time = $\frac{100}{10}$ (110 + 5)

= 1150 units.

Answer only one question on this page. Do not write on the back of this paper
Behandla endast en uppgift på detta blad. Skriv ej på baksidan

5

CHALMERS	Anonymous code / Anonym kod: DIT431-0007-ADT	Points for question (to be filled in by teacher) / Poäng på uppgiften (ifylles av lärare)	Consecutive page no. / Löpande sid nr: 2 — Question no. / Uppgift nr: 1

1b) Each task now takes 20 units, thus the execution time to context switch time ratio increases to 20:1 for ULTs and 20:5 for KLTs.

This makes using ~~N:M more suitable~~ more fine grained threading models more suitable.

~~N:1 results in execution time~~

N:1 is not suitable as no parallelism is utilised.

1:1 has execution time of 100(20+5) = 2500 units.

10:1 has 10 × (5 + 10(1+20)) = 2150 units.

100:1 has 1 × (100 (1+20)) = 2100 units

Thus 100:1 is most effective.

2.5

1c) If there are more than 1 KLT like in 1:1 or N:M, the KLTs have a shared address space.

0.5

Thus having multiple KLTs should have no issues on the program, as each KLT is able to pass the message to the different node.

(d) CUDA threads are significantly more fine grained. CUDA threads have private registers but shared memory between threads in the same block. There is also global memory that all threads can access.

There are many CUDA cores that can execute the many fine grained CUDA threads. Context switching overheads are also low as each thread has its own registers. Similar to SIMD, CUDA uses Single Instruction Multiple Threads by ~~running~~ executing the same instruction on all threads in a warp.

2.5

Answer only one question on this page. Do not write on the back of this paper
Behandla endast en uppgift på detta blad. Skriv ej på baksidan

5

CHALMERS	Anonymous code / Anonym kod	Points for question (to be filled in by teacher) / Poäng på uppgiften (fylles av lärare)	Consecutive page no. / Löpande sid nr
	DIT431-0007-ADT		3
			Question no. / Uppgift nr 2

2 a) a) Static Scheduling.

64 loop iterations. Each processor has $\frac{64}{8} = 8$ iterations.

Time units to complete $= 8 \times 2.0$
$= 16$ time units. x64

2a (b) Self-scheduling.

Each processor has 8 iterations
time units to complete $= 8 \times 2.0 = 16$ time units. x64

2a (c) Chunk-scheduling.

$\frac{64}{6} =$ ~~16 chunks.~~ 10.6 chunks

~~Each processor executes $\frac{16}{8} = 2$ chunks.~~

There are 10 chunks of size 6, 1 chunk of size 4.

6	6	4.					
6	6	6	6	6	6	6	6
P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8

Execution time $= (2 \times 6) \times 2.0 = 24$ time units. x64

2a)(d) guided-scheduling.

3

4	4	4	4	4	2		
$\frac{64}{8} = 8$	$\frac{56}{8} = 7$	$\frac{49}{8} = 6$	$\frac{43}{8} = 5$	$\frac{38}{8} = 4$	4	4	4
P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8

Execution time $= (8+4) \times 2.0 = 24$ time units.

5

CHALMERS	Anonymous code / Anonym kod	Points for question (to be filled in by teacher) / Poäng på uppgiften (ifylles av lärare)	Consecutive page no. / Löpande sid nr
	DIT431-0007-ADT		4
			Question no. / Uppgift nr 2

2b) We can ~~collasce~~ collesce (collapse) the nested loop to be of depth 1.

```
#define N 64.
for (int j=0; j<N×N ; j++) {
        int k = j % N;
        ~~[illegible]~~
        int i = j/N ;
        a[i*N+k] = (i+1)*N+k;
}
```

3

2c) Overhead > speedup.

$0.5 > ceil(\frac{x}{p}) \times 2.0$ where x is no. of iterations.

$\frac{1}{4} > ceil(\frac{4096}{p})$

$p > 1024.$

2

After $p=1024$, adding new processors result in a slowdown.

Answer only one question on this page. Do not write on the back of this paper
Behandla endast en uppgift på detta blad. Skriv ej på baksidan

5

CHALMERS	Anonymous code / Anonym kod	Points for question (to be filled in by teacher) / Poäng på uppgiften (ifylles av lärare)	Consecutive page no. / Löpande sid nr
	DIT431-0007-ADT		5
			Question no. / Uppgift nr 3

3a) $AI = \frac{FLOPs}{Bytes} = \frac{2}{4 ~~8~~ \times 4} = \frac{1}{8}$ Flops / Byte, since C has to be loaded and stored, and each ~~values~~ float is 4 bytes.

Single core: performance $= \min(4, \frac{1}{~~48~~ 8} \times 16)$
$= 2$ GFLOPs.

execution time $= \frac{100 \times 100 \times 100 \times 2}{10^6 \times 2} = 1$ s.

8 cores: performance $= \min(8 \times 4, \frac{1}{8} \times 16)$
$= 2$ GFLOPs.

execution time $= 1$ s

3b) By multiplying the performance of each core by 4, the AI is not changed thus the bottleneck is still memory. Even in a 32-core machine, the bottleneck is still the same.

3c) multi-core chip execution time $= 1$s.

upper limit $= \frac{1}{f}$ where f is serial ~~[illegible]~~ fraction

$= \frac{1}{\frac{0.5 \times 10^{-3}}{1}} = 2000.$

Using my results, since ~~a~~ the single core has the same execution time since it is bounded by memory, executing on more cores will not ~~[illegible]~~ result in more speed-up. Optimising the kernel is very suitable here. Optimising the serial portion is less suitable since the ratio of 0.5×10^{-3}s to 1s is already quite low.

5

CHALMERS	Anonymous code Anonym kod DIT431-0007-ADT	Points for question (to be filled in by teacher) Poäng på uppgiften (ifylles av lärare)	Consecutive page no. Löpande sid nr 6 Question no. Uppgift nr 4

4a) No. There is a read after write dependency. For example, when i=0 we write to A[0], but when i = N-1, we read from A[0].

1

4b).

~~paralle~~

~~#omp pra~~

```
# pragma omp parallel for ~~[illegible]~~
for (int i=0 ; i<N; i++) {
        if (i < N/2) {
            A[i] ~~[illegible]~~ += A[N-1-i];
        }
        # pragma omp barrier;
        if  (i >= N/2) {
            A[i] += A[N-1-i];
        }
}.
```

1

5

CHALMERS	Anonymous code / Anonym kod: DIT431-0007-ADT	Points for question (to be filled in by teacher) / Poäng på uppgiften (ifylles av lärare)	Consecutive page no. / Löpande sid nr: 7 · Question no. / Uppgift nr: 4

4c) quad += temp has a dependency and there is a risk that stores are lost.

First Method :
We can make quad += temp an atomic instruction using #pragma omp atomic. This way every variable can be shared, except for temp which should be private.

Second method:
We can use make quad += temp a critical region so that only 1 thread runs the instruction at a given time. We use #pragma omp critical. using shared variables, except for temp which should be private. 4

Third method:
We can use reduce in our for loop with quad and temp as shared variables. ✓
#pragma omp parallel for reduce(+: quad) private(temp)

4d) I expect Using reduce to be the fastest since every thread can still be fully run asynchronously, followed by ~~critical~~ atomic then critical, since critical requires longest waiting. For critical, compiler can create a lock for the variable quad. For atomic, a lock can also be used. For reduce, the compiler can create different quad variables for each thread.

3

5

CHALMERS	Anonymous code / Anonym kod	Points for question (to be filled in by teacher) / Poäng på uppgiften (ifylles av lärare)	Consecutive page no. / Löpande sid nr
	DIT431-0007-ADT		8
			Question no. / Uppgift nr 5

5a)

First way: We can use scatter so that the one processor can allocate the chunks of the matrix to the corresponding processors.

```
~~int id = MPI get_MPI_id;~~
MPI_Scatter( sendbuf, bytesToSend, sendType, recvbuf,
        recvCount, recvtype, 0, MPI_COMMS_WORLD);
# Now the processors portion of the matrix is in
  recvbuf
```

Second way: We can use the root to send each processors portion of the matrix.

4

```
if  (get_MPI_rank() == 0)   {
    for  (int i=1; i < P ; i++)   {
    ~~MPI_Send~~
        MPI_Send (buf[i], count, datatype, i, 0, MPI_COMMSWORLD);

    }
} else   {
                 recvbuf                   get_MPI_rank()
    MPI_Recv( ~~buf~~, count, datatype, ~~dest~~       , 0, MPI_comm_world
                                              status );
}
```

Answer only one question on this page. Do not write on the back of this paper
Behandla endast en uppgift på detta blad. Skriv ej på baksidan

CHALMERS	Anonymous code / Anonym kod	Points for question (to be filled in by teacher) / Poäng på uppgiften (ifylles av lärare)	Consecutive page no. / Löpande sid nr
5	DIT431-0007-ADT		9
			Question no. / Uppgift nr 5

5b)

```
int* block = .... // attained from 5a)
int zeros = 0;
for (int i=0; i < N/P; ~~int~~ i++) {
        for (int j=0; j<N; j++) {
                if (block[i][j] == 0) {
                        zeros++;
                }
        }
}
int* receive;

:

MPI_Reduce (&zeros, receive, P, MPI_INTEGER,
MPI_ADDITION, 0, MPI_COMMS_WORLD
                );


MPI_Bcast (receive, P, MPI_INTEGER, 0, MPI_COMMS_WORLD

                );
```

5

CHALMERS	Anonymous code / Anonym kod	Points for question (to be filled in by teacher) / Poäng på uppgiften (fylles av lärare)	Consecutive page no. / Löpande sid nr
	DIT431-0007-ADT		10
			Question no. / Uppgift nr: 5

5 c) SSend and SSrecv are synchronous and non blocking, which means SSend only returns when the other side has started receiving and SSrecv only returns when it has ~~started receiving~~ received into the buffer. This means for every SSend, it is able to synchronise with a SSrecv and vice versa, since no deadlocks were observed. Thus, even if they are replaced to non-blocking and Wait calls, no deadlock would occur.

2✓

5

CHALMERS	Anonymous code / Anonym kod: DIT431-0007-ADT	Points for question (to be filled in by teacher) / Poäng på uppgiften (ifylles av lärare)	Consecutive page no. / Löpande sid nr: 11 — Question no. / Uppgift nr: 6

6a)

A: & d_row

B: & d_col

C: & d_data

D: & d_x

E: & d_y

F: d_row, &row

G: d_col, & col

H: d_data, & data

I: d_x, & x.

7

J: 1, N.

K: d_row, d_col, d_data, d_x, d_y, N

L: &y, d_y

6b) ~~Row and colum~~

~~Data and~~

~~data and col vectors are accessed sequentially in each thread.~~

~~__global__ void spmv (.....)~~

~~__shared__ shared_~~

5

CHALMERS	Anonymous code / Anonym kod	Points for question (to be filled in by teacher) / Poäng på uppgiften (ifylles av lärare)	Consecutive page no. / Löpande sid nr
	DIT431-0007-ADT		12
			Question no. / Uppgift nr 6

6b) x offers temporal locality as each value in x is accessed multiple times. Thus we can bring x into shared memory.

```
--global-- void spmv( .... )          {
        --shared--   x_shared [N] ;
        int tid = ...
        if ( tid <N)    {
              float dot = 0.0 ;
              for (int i = row[tid] .... )   {
                    int col_ind = col[ i ] ;
                    ~~int x_val~~
                    if  (x_shared [col_ind] == -1  ) {     //(Some way
                         x_shared [col_ind] = x[col_ind] ;   of checking
                    }                                        value has not
                                                             been added)
                    dot += data [i] * x_shared [col_ind] ;
              }
              y[tid] = dot ;
        }
}
```

2

5

CHALMERS	Anonymous code / Anonym kod: DIT431 - 0007-ADT	Points for question (to be filled in by teacher) / Poäng på uppgiften (ifylles av lärare)	Consecutive page no. / Löpande sid nr: 13 — Question no. / Uppgift nr: 6

6c)

6d) Make use of loop tiling so that we access x in blocks, ~~th~~ and each thread block is able to share the block of x that they require.

2