CHALMERS
EXAMINATION / TENTAMEN

Course code/kurskod	Course name/kursnamn			
DAT400	High Performance Parallel Programming			
Anonymous code Anonym kod		Examination date Tentamensdatum	Number of pages Antal blad	Grade Betyg
DAT400-0002-LDP		03/01/2022	9	5

* 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	V	7	
2	V	10	
3	V	10	
4	V	9	
5	V	8	
6	V	8	
7			
8			
9			
10			
11			
12			
13			
14			
15			
16			
17			
Bonus poäng			
Total examination points Summa poäng på tentamen		52	

CHALMERS	Anonymous code / Anonym kod: DAT400-0002-LDP	Points for question (to be filled in by teacher) / Poäng på uppgiften (ifylles av lärare)	Consecutive page no / Löpande sid nr: 1 ; Question no. / Uppgift nr: 1

(a) To acheive minimum execution time, we need to choose optimal scheduling algorithm including how many tasks, how many iteration, how to assign the tasks to each cores (threads) and how to acheive load balancing.
those are all challenges. Also, we should consider the problem size of each iteration, if it's fixed or we can easily estimate, then static scheduling prefered. However, if it varies per iteration then dynamic scheduling is prefered.
choosing algorithm is challenging because we have to consider overhead introduced every time a task is scheduled and for example choosing parameter such as chunk size is also challenging.

(b) one of low overhead scheduling algorithms is, Static scheduling : $\frac{N}{P}$ iteration, p tasks

(N: problem size
p: # of processor)

In case of the problem size is vary,
(for example
```
    for( i=0, i<rand(k), i++){
    })
```
it cannot minimize execution time because of load-imbalance
Also, as we see on Figure 1 and Figure 2, Coarse Grained scheduling has big idleness, so in general case, it takes more time then minimum executetime.

(c) FG

without overhead : 21 x 1s = 21 s

with overhead : 21 x (1+1)s = 42s

(d) CG

without overhead : 3 x (9)s = 27 s

with overhead : 3 x (9+1) = 30 s

Without overhead, FG wins

with overhead CG wins

2

CHALMERS	Anonymous code / Anonym kod: DAT400-0002-LDP	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: 2

2. $N=64$ $p=8$

(a) 1. $\frac{64}{8}$ = 8 iterations, 8 tasks

2. 1 iteraton, 64 tasks

3. $\lceil \frac{64}{6} \rceil = 11$, 6 iterations × 10 tasks, 4 iteraton × 1 task, 11 tasks

4. GSS(4) = 13 tasks, each iteratons $\underbrace{8, 7, 7, 6, 5, 4, 4, 4}_{31}$, 4, 4, 4, 4, 4, ~~4~~

~~4×8~~

8

C0	8	
C1	7	
C2	7	
C3	6	4
C4	5	4
C5	4	4
C6	4	4
C7	4	4

GSS(4)

37

7

Texe = 2.
4

(b)

1. 8 × 2 = 16 s

2. $\frac{64}{8}$ × 2 = 16 s

3. 2 × 6 × 2 = 24 s

4. 2 × 10 = 20 s (on C3, 10 iteratons)

(c) self-scheduling
8 × (2+0.2) = 17.6 s

chunk scheduling

2 × (2×6 + 1) = 26 s

Eventhoug there is overhead on self scheduling, because of load imbalance on chunk scheduling, Scheduler 2 (self-scheduling) is faster.

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: DAT400-0002-LDP	Points for question (to be filled in by teacher) / Poäng på uppgiften (ifylles av lärare)	Consecutive page no. / Löpande sid nr: 3 — Question no. / Uppgift nr: 3

$\frac{3}{Q}$

		1	2	3	4	5	6	7	8	9	10	11	12	13	14
2cores															
Case1	C0	1	2	13	5	6	15	9	10	17	$\boxed{19}$	$\to$ 22 $\to$ 23			
	C1	3	4	14	7	8	16	11	12	18	$\boxed{20}$	$\boxed{21}$			
Case2	C0	1	3	5	7	9	11	13	15	17	19 $\to$	21 $\to$	23		
	C1	2	4	6	8	10	12	14	16	18	20 $\to$	$\boxed{22}$			

$\to$ 12 time units

4cores		1	2	3	4	5	6	7	8	9
case1	C0	1	5	9	13	17	$\boxed{21}$	23		
	C1	2	6	10	14	18	$\boxed{22}$			
	C2	3	7	11	15	19				
	C3	4	8	12	16	20				

$\to$ 7 time units

C0	1	2	3	4	
C1	5	6	7	8	..
C2	9	10	11	12	$\boxed{x}$
C3	x	x	13	14	..

C0	1	2	9	10	17	$\boxed{21}$	23
C1	3	4	11	12	18	$\boxed{22}$	
C2	5	6	13	14	19		
C3	7	8	15	16	20		

8 cores	1	2	3	4	5	6	7
C0	1	9	17 $\to$	21 $\to$	23		
C1	2	10	18	22			
C2	3	11	19				
C3	4	12	20				
C4	5	13					
C5	6	14					
C6	7	15					
C7	8	16					

$\to$ 5 time units.

CHALMERS	Anonymous code / Anonym kod: DAT400-0002-LDP	Points for question (to be filled in by teacher) / Poäng på uppgiften (ifylles av lärare)	Consecutive page no / Löpande sid nr: 4
			Question no. / Uppgift nr: 3

3 (b)

Cost = $p \times T_{exe}(n)$, p = ~~#~~ # of cores, $T_{exe}(n)$ parallel execution time

$$\text{Efficiency} = \frac{\text{The best sequantial exe time}}{\text{Cost}}$$

2 cores : cost = $2 \times 12 = 24$ tu. Eff : $\frac{16}{24} = 0.667$

4 cores : cost = $4 \times 7 = 28$ tu Eff : $\frac{16}{28} = 0.571$

8 cores : cost = $8 \times 5 = 40$ tu Eff : $\frac{16}{40} = 0.4$

(c) the largest number of cores that can accelerate the computation is 12, because at the first level of graph (task 1 ... 12), we can process maximum 12 tasks at the same iteraton time.

Critial path is 5.

(d) For each task, we can ~~d~~ apply fine grain schedule which devide the task even smaller, in this case we should carefully consider overhead.

the other approach is that we ~~ch~~ can use caches so that reduce memory access time which leads reduced execute time and better performance.

2

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
	DAT400-0002-LDP		Question no. / Uppgift nr 4

(a)

N^2(1sub+1sub + get_invDist3 +1mul + 1add + 1mul + 1add)

+N (1mul + 1add + 1mul + 1add)

= by assuming that sub, mul, add take 1 FP operation

= N^2(6+get_invDist3)+4N

data / loop level parallelism possible. Because there's no data dependency.

(b) ~~pri~~ private (j)

shared (i , Fx, Fy, N)

(c) there are race ~~co~~ condition when update Fx, Fy (line 13, 14) since multiple threads access this variable.

To solve this, we should set critical section, atomic ~~to~~ or reduction

by adding

(above ~~9~~ line 9) # pragma omp parallel for

(above line 13) # pragma omp critical

or (above line 13) # pragma omp atomic

or (abov line 9) # pragma omp paraller for reduction(+: Fx), reduction (+: Fy)

OR another approach is that privitization of Fx, Fy but there might be false sharing problem.

(d) I prefer outerloop (target loop) to paralelize since, if we paralelize inner loop, it creates ~~N~~ and join N threads for the iterations, which occurs ~~big~~ big overhead.

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
	DAT400-0002-LDP		6
			Question no. / Uppgift nr 5

(a) tree shape Implementaion Is better. Because when headnode ~~MPI_Send()~~ MPI_Bcast() then the other node can recive. so It's effective way when it comes to efficiency.

(b)
```
/ measue exe time for group 1 broadcastig /
MPI_Barrier ( )
measuredtime -= MPI_Wtime ( )
MPI_Bcost (        ...          , comm_WORLD1)
MPI_Barrier( )                  // Bradcast to
measuredtime += MPI_Wtime ( )      group1

/ mesure exe time for group 2 broadcastig /
MPI_Barrier ( )
measured time -= MPI_Wtime ( )
MPI -Bcast (     ...       , comm_WORLD_2 )
MPI _Barrier ( )
measuredtie += MPI_Wtime( )
```

(c)
```
myrank = MPI _ Rank
msize = MPI - Size
MPI_Scatter (                                  ) }

if ( myrank==0) }
    for ( irank=0 ; irank < msize  ++ irank ) }
        MPI_Send( buf[ irank*m ] , sizeof(flow) , 0 ,   )
                                                  tag

}   MPI_Recv( buf[  ] , sizeof(flat) , 0 , root , comm_   )
                                      tag
```

2

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
	DAT400-0002-LDP		1
			Question no. / Uppgift nr 5

ⓒ

```
MPI_Scatter( buf,    sizeof(float), rank   , tag,        ,        )5

    myrank = ~~MPI_Rank~~
         Comm_Rank (        ,&    )

    ~~my~~
    msize= Comm_size (      , &   )

    if(myrank==0) {
        for(irank=0 ;  irank < msize  ++ irank) {
                    irank+msize* sizeof(float)
        MPI_Send( buf[              ]  ,  irank,  0   ,              )
                      ~~sizeof(float~~

        }

    MPI_Recv(Buf[          ]  ,          , root,  0 ,         )
                  ~~sizeof float~~

}


 }
```

2

CHALMERS	Anonymous code / Anonym kod DAT 400-0002 - LDP	Points for question (to be filled in by teacher) / Poäng på uppgiften (fylles av lärare)	Consecutive page no / Löpande sid nr 8 / Question no. / Uppgift nr 6

ⓐ
1) BLOCK_DIM   for example 1024, (in this case, blockDim.x)
2) (n-1)/BLOCK_DIM +1   "   8 , ((n-1)/BLOCK_SIZE +1)
3) *d_input, *input
4) *output, *d_output

ⓑ line 5, the result overwrites in input.
we should add ~~sum +~~ sum += input[i] at line 6.

Also, after ~~ad~~ calculaton, we shoud add
syncthreads(); make sure the computation finised.

sizeof(float) x vectorlength x N

32

Bytes transfered from host to device: 4 x 1 x 8192 = 32KB

ⓒ
total executon time of the code
: exe time of kernel functon + exetime of squential code + communicaton from Host to device + communication from device to Host

$= 10S + 2S + 2\times\left(\frac{32KB}{256GB/s}\right)$

$\fallingdotseq 12S$

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

2

CHALMERS | Anonymous code / Anonym kod: DAT400-0002-LDP | Points for question (to be filled in by teacher) / Poäng på uppgiften (ifylles av lärare) | Consecutive page no. / Löpande sid nr: 9 | Question no. / Uppgift nr: 6

ⓓ using shared memory with GPU architectue.

as we see in ©, BW is big enough.

```
BLOCK_DIM = 1024;
__global__ void Reduce 1 (                    ) {
    unsigned int i = blockDim.x * blockIdx.x + threadIdx.x;
    __shared__ float sum[BLOCK_DIM];
    for(unsigned int stride = 1; stride < blockDim.x; stride *= 2)) {
        if (threadId.x % (stride * 2))) == 0)
            ~~input~~
            input[i] += input[i + stride];
            sum[tid] = input[ BLOCK_DIM * i + tid];

    }

    if (threadId.x == 0) {

        :
        :
        :
```

Same with Reduce 0