NHR FAU

FAU Friedrich-Alexander-Universität Erlangen-Nürnberg

# Programming Techniques for Supercomputers

Erlangen National High Performance Computing Center

Department of Computer Science

FAU Erlangen-Nürnberg

Sommersemester 2024

# Assignment 5 – Task 1

```
$ likwid-topology -g
--------------------------------------------------------------------------------
CPU name:       Intel(R) Xeon(R) Platinum 8470
CPU type:       Intel SapphireRapids processor
CPU stepping:   8
********************************************************************************
Hardware Thread Topology
********************************************************************************
Sockets:                2
Cores per socket:       52
Threads per core:       1
--------------------------------------------------------------------------------
HWThread        Thread          Core        Die         Socket          Available
0               0               0           0           0               *
1               0               1           0           0               *
2               0               2           0           0               *
3               0               3           0           0               *
4               0               4           0           0               *
5               0               5           0           0               *
6               0               6           0           0               *
7               0               7           0           0               *
[...SNIP...]
102             0               102         0           1               *
103             0               103         0           1               *
--------------------------------------------------------------------------------
```

2 sockets,
104 cores/node

# Assignment 5 – Task 1

```
$ likwid-topology -g
[...SNIP...]
********************************************************************************
Cache Topology
********************************************************************************
Level:                  1
Size:                   48 kB
Cache groups:           ( 0 ) ( 1 ) ( 2 ) ( 3 ) ( 4 ) ( 5 ) ( 6 ) ( 7 ) ( 8 ) ( 9 ) ( 10 ) ( 11 ) ( 12 ) ( 13 ) ( 14 ) (
15 ) ( 16 ) ( 17 ) ( 18 ) ( 19 ) ( 20 ) ( 21 ) ( 22 ) ( 23 ) ( 24 ) ( 25 ) ( 26 ) ( 27 ) ( 28 ) ( 29 ) ( 30 ) ( 31 ) ( 32
) ( 33 ) [...SNIP...] ( 74 ) ( 75 ) ( 76 ) ( 77 ) ( 78 ) ( 79 ) ( 80 ) ( 81 ) ( 82 ) ( 83 ) ( 84 ) ( 85 ) ( 86 ) ( 87 ) (
88 ) ( 89 ) ( 90 ) ( 91 ) ( 92 ) ( 93 ) ( 94 ) ( 95 ) ( 96 ) ( 97 ) ( 98 ) ( 99 ) ( 100 ) ( 101 ) ( 102 ) ( 103 )
--------------------------------------------------------------------------------
Level:                  2
Size:                   2 MB
Cache groups:           ( 0 ) ( 1 ) ( 2 ) ( 3 ) ( 4 ) ( 5 ) ( 6 ) ( 7 ) ( 8 ) ( 9 ) ( 10 ) ( 11 ) ( 12 ) ( 13 ) ( 14 ) (
15 ) ( 16 ) ( 17 ) ( 18 ) ( 19 ) ( 20 ) ( 21 ) ( 22 ) ( 23 ) ( 24 ) ( 25 ) ( 26 ) ( 27 ) ( 28 ) ( 29 ) ( 30 ) ( 31 ) ( 32
) ( 33 ) [...SNIP...] ( 74 ) ( 75 ) ( 76 ) ( 77 ) ( 78 ) ( 79 ) ( 80 ) ( 81 ) ( 82 ) ( 83 ) ( 84 ) ( 85 ) ( 86 ) ( 87 ) (
88 ) ( 89 ) ( 90 ) ( 91 ) ( 92 ) ( 93 ) ( 94 ) ( 95 ) ( 96 ) ( 97 ) ( 98 ) ( 99 ) ( 100 ) ( 101 )
--------------------------------------------------------------------------------
Level:                  3
Size:                   105 MB
Cache groups:           ( 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34
35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 ) ( 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73
74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100 101 102 103 )
--------------------------------------------------------------------------------
[...SNIP...]
```

Per-core L1 (48 KiB) and L2 (2 MiB) caches

Socket-shared L3 (105 MiB) cache

# Assignment 5 – Task 1

```
$ likwid-topology -g
********************************************************************************
NUMA Topology
********************************************************************************
NUMA domains:             8
--------------------------------------------------------------------------------
Domain:                   0
Processors:               ( 0 1 2 3 4 5 6 7 8 9 10 11 12 )
Distances:                10 12 12 12 21 21 21 21
Free memory:              128272 MB
Total memory:             128739 MB
--------------------------------------------------------------------------------
[...SNIP...]
--------------------------------------------------------------------------------
Domain:                   7
Processors:               ( 91 92 93 94 95 96 97 98 99 100 101 102 103 )
Distances:                21 21 21 21 12 12 12 10
Free memory:              128761 MB
Total memory:             129017 MB
--------------------------------------------------------------------------------

--------------------------------------------------------------------------------
```

4 ccNUMA domains per socket (13 cores)

# Assignment 5 – Task 2

```
double t_start,t_end;
int id, nt, k, I, NITER, N;
double t = 0;
double *a = (double*)malloc(N*sizeof(double));
double *b = (double*)malloc(N*sizeof(double));
double *c = (double*)malloc(N*sizeof(double));

#pragma omp parallel for
for(k=0; k<N; ++k)
  a[k]=b[k]=c[k]=0.;

NITER=1;
do {
  t_start = getTimeStamp();

  for(k=0; k<NITER; ++k) {
    #pragma omp parallel for
    for(i=0; i<N; ++i) {
      a[i] = b[i] + 1.0000001 * c[i];
    }
    if(a[N/2]<0.) printf("%lf",a[N/2]);
  }
  t_end = getTimeStamp();
  NITER = NITER*2;
} while (wct_end-wct_start<1.);

NITER = NITER/2;

printf("Total walltime: %f, NITER: %d, Perf: %.3lf GF/s\n",
t_end-t_start, NITER, 2.*(double)N*NITER/(t_end-t_start)*1.e-9);
```

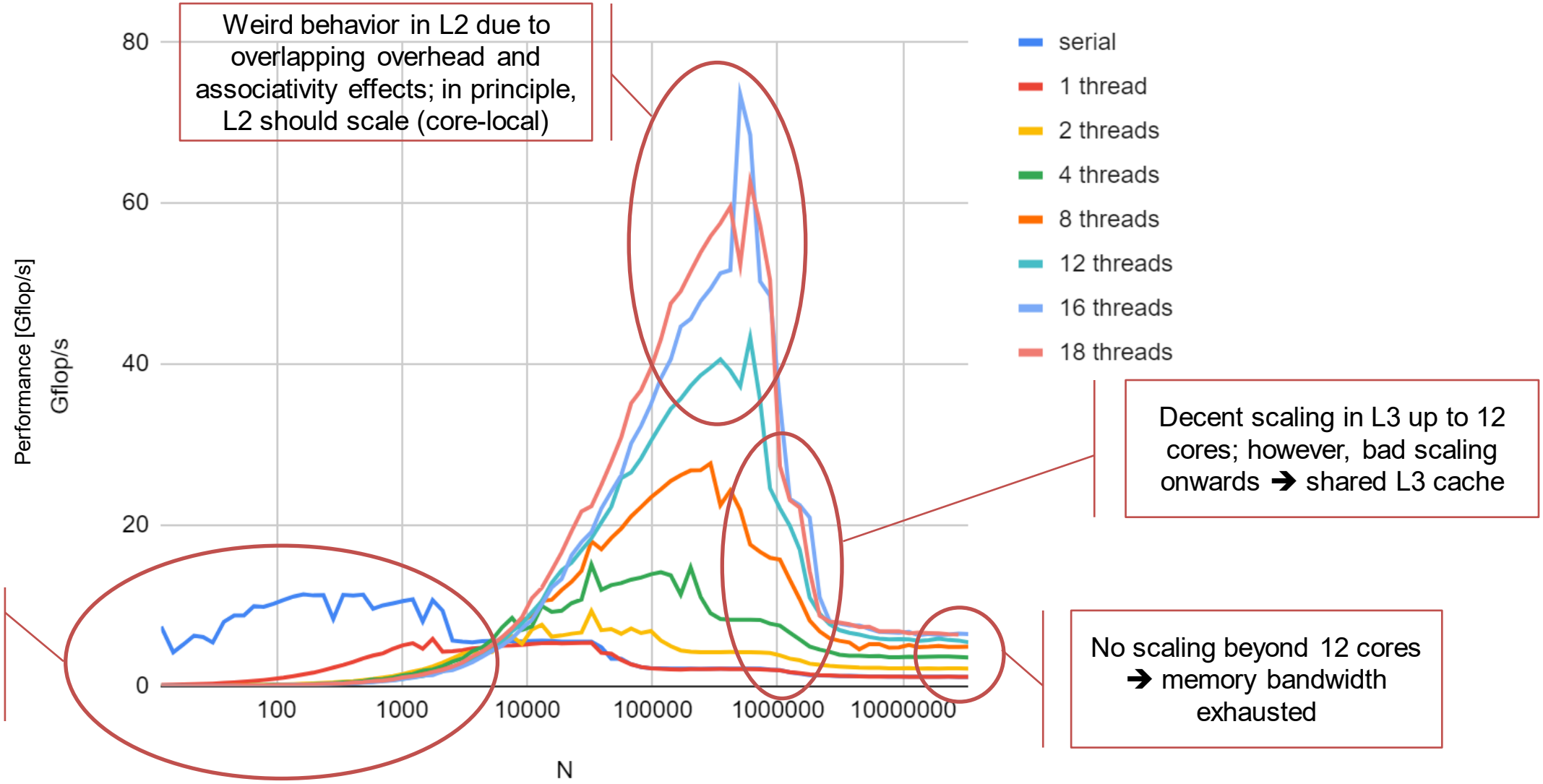
```
icx -O3 -xHost -qopenmp
scan_c.c timing.c
```

# Assignment 5 – Task 2

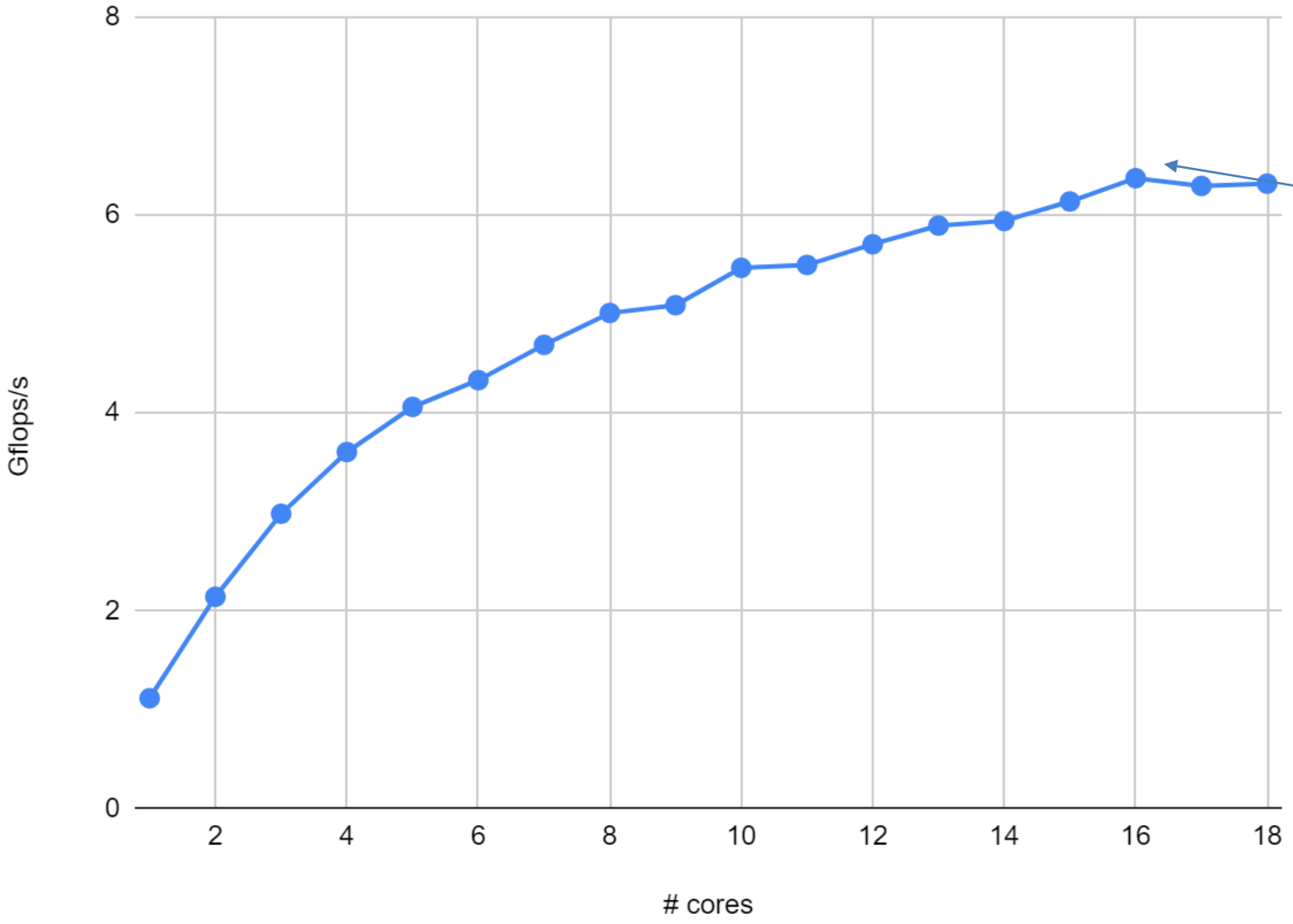
```
#/bin/bash
for t in 1 2 4 8 16 18; do
    n=10
    for i in `seq 1 90`; do
        echo -n $n " "
        srun -cpu-freq=2000000-2000000 likwid-pin -c 0-$(( t - 1 )) -q \
                                       ./a.out $n >> update_${t}.out
        n=$(( (n * 12) / 10 ))
    done
done
```

```
#/bin/bash
for t in 1 2 4 8 16 18; do
    n=10
    for i in `seq 1 90`; do
        echo -n $n " "
        OMP_NUM_THREADS=$t OMP_PLACES=cores OMP_PROC_BIND=close \
                           srun -cpu-freq=2000000-2000000 \
                           ./a.out $n >> update_${t}.out
        n=$(( (n * 12) / 10 ))
    done
done
```

# Assignment 5 – Task 2



# Assignment 5 – Task 2




```
#pragma omp parallel for
for (int i=0; i<N; ++i)
    a[i] = b[i] + s * c[i];
```

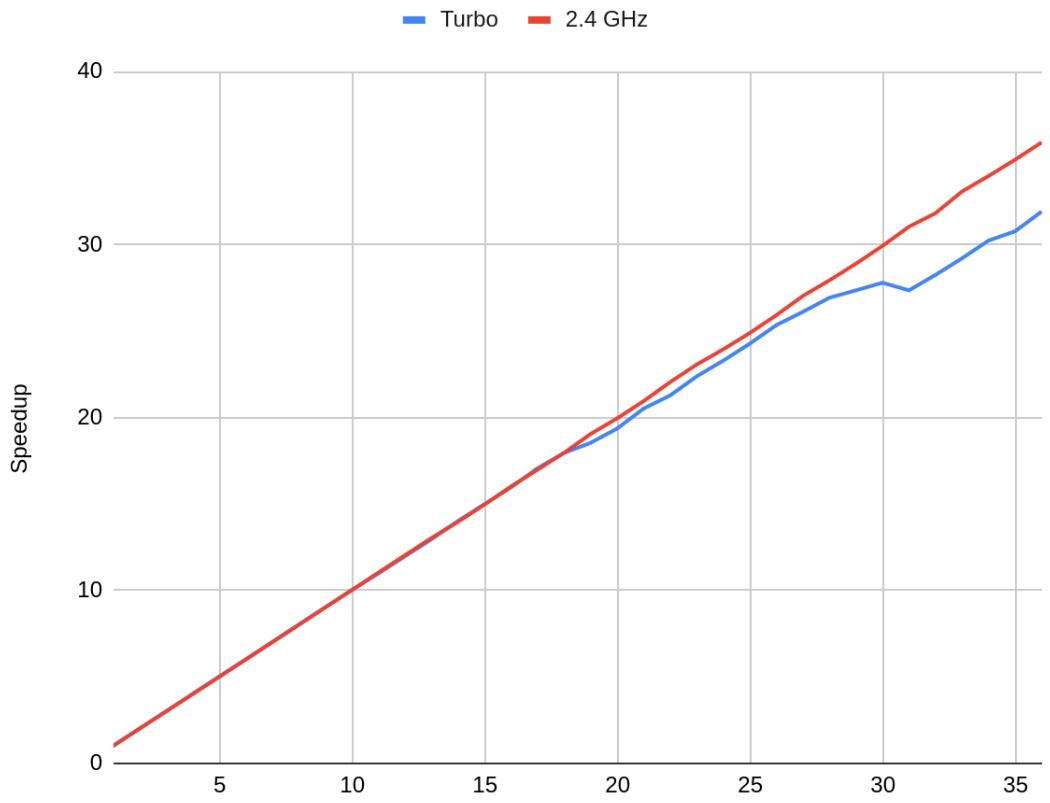
$$b = 6300 \frac{MFLOP}{s} \times 16 \frac{B}{FLOP} = 100.8 \frac{GB}{s}$$

This is just a little too close to the theoretical limit of 102.4 GB/s → Write-Allocate Evasion! Better estimate:

$$b = 6300 \frac{MFLOP}{s} \times 12 \frac{B}{FLOP} = 75.6 \frac{GB}{s}$$

The truth is somewhere in between.

# Assignment 5 – Task 3




```
// bogus parallel region
#pragma omp parallel
  if(omp_get_thread_num()==0) printf("Hello");

  S = getTimeStamp();
#pragma omp parallel for reduction(+:sum) private(x)
  for(int i=0; i<N; i++) {
    x = (i + 0.5) * delta_x;
    sum = sum + 4.0 * sqrt(1.0 - x * x);
  }
  E = getTimeStamp();
```

```
$ OMP_NUM_THREADS=$t OMP_PLACES=cores \
  OMP_PROC_BIND=close \
  srun --cpu-freq=performance ./a.out
```

- Perfect scaling with fixed frequency!
- Degraded scaling with Turbo Mode starting @19 cores → clock speed reduction!