Sheet 09

PS Parallel Programming

Patrick Wintner

May 19, 2025

1 Compiler Dependence Analysis

The dependence analysis of compilers is examined.

1.1 Source Code

1.1.1 Original

```c
#pragma omp parallel default(none) shared(n, a, b, c, local_res)
	{
		// matrix multiplication
#pragma omp parallel for default(none) shared(n, a, b, c)
		for (long i = 0; i < n; ++i) {
			for (long j = 0; j < n; ++j) {
				for (long k = 0; k < n; ++k) {
					c[i][j] += a[i][k] * b[k][j];
				}
			}
		}

		// sum of matrix c
#pragma omp parallel for default(none) shared(n, a, b, c, local_res)
		for (long i = 0; i < n; ++i) {
			for (long j = 0; j < n; ++j) {
				local_res[omp_get_thread_num()] += c[i][j];
```

```
				}
			}
		}
		unsigned long res = 0;
		for (int l = 0; l < omp_get_num_threads(); ++l) {
			res += local_res[l];
		}
```

1.1.2 Improved

```
#pragma omp parallel default(none) shared(n, a, b, c, local_res)
		{
			// matrix multiplication
#pragma omp parallel for default(none) shared(n, a, b, c)
			for (long i = 0; i < n; ++i) {
				for (long j = 0; j < n; ++j) {
					for (long k = 0; k < n; ++k) {
						c[i][j] += a[i][k] * b[k][j];
					}
				}
			}

			// sum of matrix c
#pragma omp parallel for default(none) shared(n, a, b, c, local_res)
			for (long i = 0; i < n; ++i) {
				for (long j = 0; j < n; ++j) {
					local_res[omp_get_thread_num()] += c[i][j];
				}
			}
		}
		unsigned long res = 0;
		for (int l = 0; l < omp_get_num_threads(); ++l) {
			res += local_res[l];
		}
```

1.2 Measurement Method

All measurements were done on the LCC3 cluster by calling `sbatch job.sh <executable> 3`, e. g. `sbatch job.sh original 3` (3 is the number of measurements) with the size of the matrix set to 1500.

The following scripts are involved in the experiment.

1.2.1 SLURM Script

```
#!/bin/bash
# usage: sbatch <executable> <number_of_measurements>

# Execute job in the partition "lva" unless you have special requirements.
#SBATCH --partition=lva
# Name your job to be able to identify it later
#SBATCH --job-name csba4017
# Redirect output stream to this file
#SBATCH --output=output.log
# Maximum number of tasks (=processes) to start in total
#SBATCH --ntasks=1
# Maximum number of tasks (=processes) to start per node
#SBATCH --ntasks-per-node=1
# Enforce exclusive node allocation, do not share with other jobs
#SBATCH --exclusive

./benchmark.sh $1 $2
```

1.2.2 Benchmark Script

```
#!/bin/bash
# Usage: ./benchmark.sh <executable> <number_of_measurements>

N=100 # size of matrix
results=$1".dat"
echo "x y ey" > $results # create header
make toTable
make $1
for i in {1,2,4,6,12} # number of threads
do
        measurements=$i"_"$1".log"
        export OMP_NUM_THREADS=$i
        for j in $(seq 1 $2) # repeat measurement £2 times
        do
                ./$1 $N >> $measurements # store measurement results in <executable>.log
        done
```

```
        ./toTable $measurements $results $2 $i #store table in <executable>.dat
        rm $measurements
done
make clean
```

1.3 Experiment Results