OpenMP lab

1 Task 1

Implement the algorithm for approximating π as given in the lecture in C. Expose concurrency using openMP. Do not use the openMP support for loops. Do not use the mutual exclusion support (neither *critical* nor *atomic*). Measure the runtime performance for varying numbers of threads and relate that to the number of cores of the hardware that you use.

2 Task 2

Try to avoid the false sharing of the solution given in the lecture by turning the array `sum` into a two dimensional array with [10][64] elements. Instead of using `sum[id]` do use `sum[id][0]` for computing the partial sums for each thread. Measure the runtime impact of this modification. Why/how does that work?

3 Task 3

As explained in the lecture, use openMP's mutual exclusion support for avoiding the array `sum`. How do you need to use it to achieve the best possible performance?

4 Task 4

Use the loop support of openMP to further simplify your implementation of the π approximation. Investigate the impact of different schedulings. Which one gives you the best performance results?

Hand-out: 18/02/2015