Technische Universität Kaiserslautern
Fachbereich Informatik
RHRK
Priv.–Doz. Dr. Josef Schüle

Introduction in High Performance Computing
Exercise Sheet 4

Parallelization of CG Algorithm

The Conjugate Gradient (CG) Algorithm is a very powerful algorithm to solve linear equation systems

$$Ax = b$$

for symmetric sparse matrices A. The basic algorithm, starting in line 64 of the file `ssolo.c`, consists of a series of vector operations (`sscal, saxpy`), a dense matrix-vector multiplication `sgemv` and the formation of the scalar product `sdot` between two vectors:

```
sscal(n, beta, p, 1);
saxpy(n, 1.0f, r, 1, p, 1);
saxpy(n, alpha, p, 1, x, 1);
sscal(n, beta, s, 1);
saxpy(n, 1.0f, q, 1, s, 1);
saxpy(n, (-1)*alpha, s, 1, r, 1);
sgemv(\"N\", n, n, 1.0f, A, n, r, 1, 0.f, q, 1);
beta = rho;
rho = sdot(n, r, 1, r, 1);
mu = sdot(n, q, 1, r, 1);
beta = rho / beta;
alpha = rho/(mu-rho*beta/alpha);
rnorm = rho;
```

All of these functions obey the calling sequence of the BLAS library with `alpha, beta` being (scalar) values, `A` being an $n \times n$ matrix with first dimension $lda = n$ and `p, q, r, s` being vectors of length n. In the calling sequence these vectors are followed by the increment they are accessed ($= 1$), that is, the call of `sscal(n,beta,p,1)` is equivalent to the following loop:

```
for(i=0;i<n;i++) p[i]*=beta;
```

Your task is to

- analyze the code with help of vtune and insert vectorization
- parallelize the code with OpenMP (hint: start with the expensivst function, ...

Don't forget the overhead for a `parallel for` or a `parallel` pragma and consider to set up the team of threads not for every loop. Run the OpenMP version on 1,...,16 cores with $n = 20,000$.
Does the program scale going from 4 to 16 cores according the $O(n^2)$ complexity, that is, compare $n = 20,000$ with 4 cores to $n = 40,000$ with 16 cores with FIXED number of iterations?

OpenMP Benchmark - Bonus work

Search for the OpenMP Microbenchmark and download its newest version (see lecture). Compile the Benchmark with the Intel Compiler and perform the syncbench and schedbench Tests for 1 to 16 cores. Use LSF to submit the tests to a node allocated for you only. Select 4 different tests of the 2 suites and depict and interpret the results (best, if you select different tests than your neighbor).