Università della Svizzera italiana

Institute of Computational Science ICS

High Performance Computing **2018**

Instructor: Prof. Olaf Schenk TA: Juraj Kardos, Radim Janalik

---

Assignment 8

Due date: 19 December 2018, 13:30

---

1. Measuring the Memory Bandwidth *(10 Points)*

STREAM is a simple, synthetic benchmark designed to measure sustainable memory bandwidth (in MB/s) and a corresponding computation rate for simple vector kernels. Like the LINPACK benchmark, this is intended to show off the best possible bandwidth of the large HPC systems. The STREAM benchmark `bench.c` implements four kernels to probe the available memory bandwidth. These are copy (`a[i]=b[i]`), scale (`a[i]=alpha*b[i]`), add (`a[i]=b[i]+c[i]`) and triad (`a[i]=b[i]+alpha*c[i]`). Use the copy benchmark to measure the available memory bandwidth of the compute node, which should approximate the upper limit. Run the benchmark with 1-20 threads to see when does the memory bandwidth become saturated. The result for the vector triad benchmark is shown in figure 1. Produce a similar figure.

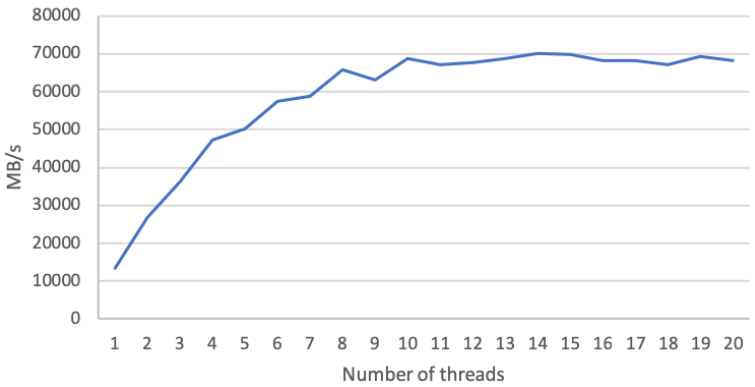

Figure 1. Vector triad benchmark for the ICS compute node.

2. Estimating Single Core Performance of Vector Triad Benchmark *(40 Points)*

On a compute node of the *icsmaster* cluster there are two Intel Xeon E5-2650 v3 CPUs (Haswell[1] architecture). Each CPU has 10 cores with frequency 2.30 GHz and both CPUs share 64 GB RAM memory. Haswell architecture supports AVX2 instructions. This allows combining Advanced Vector Extensions (AVX) and Fused Multiply–Add (FMA) operations. AVX performs operations on 256-bit registers. Each register contains 8 single precision or 4 double precision floating points numbers. Figure 2 shows detailed information about the Haswell architecture.

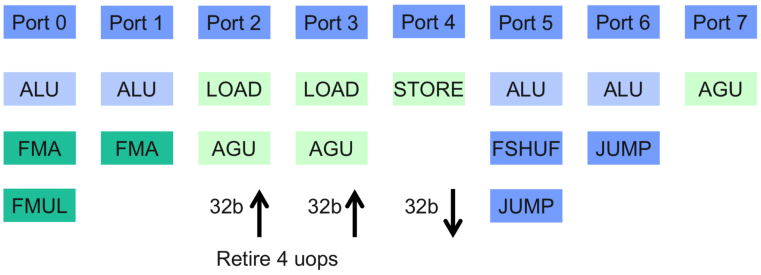

Figure 2. Architecture of Haswell core.

Figure 3 shows measured (black line) and estimated (pink line) **single core** performance of the vector triad benchmark on Intel Sandy Bridge processor. Using the information above and in the lecture slides, compute the expected **double precision** performance of the four benchmarks at the *icsmaster* compute node, when AVX and FMA instructions are used. The four benchmarks are:

- dot product: `res += a[i]*b[i]`
- daxpy: `y[i]=y[i]+a*x[i]`
- stream: `y[i] = x[i]+c*z[i]`
- triad: `y[i]=x[i]+c[i]*d[i]`.

After computing the theoretical performance, use the `likwid-bench` tool to measure the performance for different vector lengths (see instructions below) to verify your estimate.

[1] Intel Xeon Processor E5-2650 v3: https://ark.intel.com/products/81705/Intel-Xeon-Processor-E5-2650-v3-25M-Cache-2-30-GHz-

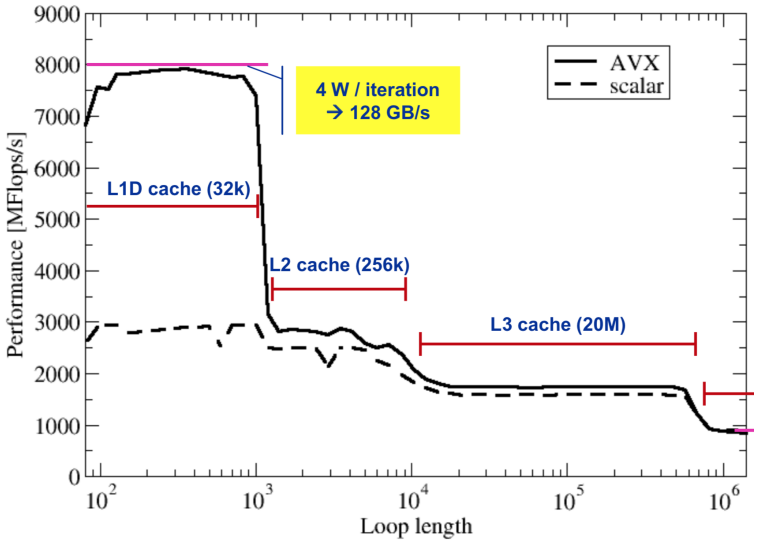

Figure 3. Performance of vector triad benchmark on 1 core of Intel Sandy Bridge processor.

Before using likwid, load the module

```
$ module load likwid
```

Print all available tests

```
$ likwid-bench -a
```

Run a menchmark

```
$ likwid-bench -t <TEST> -w <thread_domain>:<size>:<num_threads>
```

Choose the appropriate benchmark in double precision optimized for AVX and FMA (e.g. triad_avx_fma). Use the domain S0 and 1 thread. Try different sizes such way that the data fit in L1, L2, L3 cache and main memory, for example 16 kB, 128 kB, 10 MB and 10 GB. Plot your measurement results for the benchmarks, producing similar figure to 3. Compare your expectation with the measurement for L1 cache. Is your expectation reasonable?

Note: Make sure you are running the benchmark on compute node and that you are the only user on that node.

3. The Roofline Model *(20 Points)*

Draw a roofline model (for 1 and 10 cores) for the compute node considering the CPU specific parameters listed in section 2 and the appropriate memory bandwidth measured in section 1. Clearly indicate the titles and units of both x and y axes. Also clearly specify the values of the x and y ticks.

Answer the following questions:

- For which arithmetic intensity can we expect memory bound applications?

- What is the minimum arithmetic intensity for which the computation is bounded by the peak performance of the CPU?

4. Analysis of the Code Using The Roofline Model *(30 Points)*

```
double x[N], y[N], z[N], res, a;

// Kernel A - norm^2 of the vector
#pragma omp parallel for reduction(+:res)
for (int i = 0; i < N; i++)
{
   res += x[i] * x[i];
}

// Kernel B - dot product
#pragma omp parallel for reduction(+:res)
for (int i = 0; i < N; i++)
{
    res += x[i] * y[i];
}

// Kernel C - vector operation
#pragma omp parallel for
for (int i = 0; i < N; i++)
{
    z[i] += a*x[i] + y[i];
}
```

Assume all possible optimizations (for example loop unrolling). Answer the following questions and provide all the computations including units.

- Compute the arithmetic intensity and code balance for all kernels.
- Indicate the position of the kernels in the Roofline model from previous section. Consider two scenarios, (i) `OMP_NUM_THREADS=1` and (ii) `OMP_NUM_THREADS=10`.
- What is the expected performance of the kernels? Which kernels are limited by the peak performance and which by memory bandwidth?

Submission:

Submit the source code files in an archive file (tar, zip, etc) and show the TA the results. Furthermore, summarize your results and the obvervations for all exercises by writing an extended Latex summary. Use the Latex template from the webpage and upload the extended Latex summary as a PDF to iCorsi.