COLFAX RESEARCH | SOFTWARE EXPERTISE EVALUATION (SEE)

Vectorizing Monte-Carlo Diffusion

Download lab file

File list:

- Makefile
- main.cc
- diffusion.cc*
- distribution.h*
- distribution.cc*

(Only the changes in * files are used for grading)

Instructions:

In this lab, you will be working with an application simulating 1-D random walk. We have a large number of particles starting at the center (x=0), and at each time step the particles are allowed to move randomly according to a given distribution function. The goal is to find the number of particles that have position (x) greater than a certain threshold

Distribution function can be found in "distribution.cc" and "distribution.h".

```
 const float delta_max = 1.0f;
float dist_func(const float alpha, const float rn) {
  return delta_max*sinf(alpha*rn)*expf(-rn*rn);
}
```

Here, *rn* is a randomly generated number

The distribution function is used in simulating diffusion in the "diffusion.cc" file.

```
int diffusion(
     const int n_particles,     // num of particles
     const int n_steps,         // num of timesteps
     const float x_threshold,   // x cutoff
     const float alpha,         // for dist_func
     VSLStreamStatePtr rnStream // RNG
    ) {
  int n_escaped=0;
  for (int i = 0; i < n_particles; i++) {
    float x = 0.0f;
    for (int j = 0; j < n_steps; j++) {
      float rn;
      // Intel(R) MKL RNG
      vsRngUniform(VSL_RNG_METHOD_UNIFORM_STD,
                   rnStream, 1, &rn, -1.0, 1.0);
      x += dist_func(alpha, rn);
    }
    if (x > x_threshold) n_escaped++;
  }
  return n_escaped;
}
```

Unfortunately, the diffusion workload starts unvectorized. Modify "distribution.cc", "distribution.h" and "diffusion.cc" so that auto-vectorizer can vectorize diffusion().

Hints:

We recommend approcahing this assignment in the following two steps.

Step 1: vectorizing dist_func()

Add either a c++ attribute or OpenMP pragma to instruct the compiler produce a vectorized version of the function.

Step 2: vectorizing diffusion()

The diffusion workload has a true vector dependency over timesteps, so this workload can not be vectorized over timesteps. One solution to this issue is to implement loop-interchange to make the particle loop to the inner loop.

In order to implement the interchange, you must create a temporary buffer to store the positions of the particles. Furthermore, the random number generator can't be in the vectorized loop. So you must generate and store multiple random numbers before the vectorized loop. You can generate *n_particles* random numbers with:

```
float rn[n_particles];
vsRngUniform(VSL_RNG_METHOD_UNIFORM_STD,
             rnStream, n_particles, rn, -1.0, 1.0);
```

You need to add an appropiate pragma to have the compiler use vectorized version of dist_func()

Rules:

There are few rules for this lab:

- Do NOT use explicit vectorization for this assignment. The grading script specifically checks for auto vectorization.
- Do not add multi-threading for this assignment. The grading script limits the application to a single core.
- Do not change the name of the distribution function "dist_func()". The grading script specifically looks for that function name.

Running app:

The grading script uses the following command to run the application.

```
% taskset -c 0 ./app $ALPHA $THRESHOLD
```

Grading:

1. **Compile:** The code compiles without error (0 points)
2. **Verification:** The code generates the correct output (0 points)
3. **Vectorized dist_func():** dist_func() in "distribution.cc" is vectorized (0.3 points)
4. **Vectorized diffusion():** diffusion() in "diffusion.cc" is vectorized (0.3 points)
5. **Performance:** The application completes in under 2.1 seconds on the Xeon Phi 7210 system (0.4 points)

Last transaction TID: 29124
Your grade has been sent to the course platform.

Solution: 选取文件 未选择文件

Resubmit

TID	Grade	Feedback	Submitted file	Submission time
29124	100 *	Compilation: PASSED Verification: PASSED Vectorized dist_func(): PASSED Vectorized diffusion(): PASSED Performance: PASSED	Submitted solution	2020/12/16 11:09:42
29123	0	Compilation: FAILED make command did not produce the executable and/or the optrpt files	Submitted solution	2020/12/16 11:07:37
29122	30	Compilation: PASSED Verification: PASSED Vectorized dist_func(): PASSED Vectorized diffusion(): FAILED diffusion() does not appear to be vectorized	Submitted solution	2020/12/16 10:51:04

Legend:

* - your best grade for this lab so far

x - not yet submitted to the course platform. Grade transfer may take up to 5 minutes. To re-check the status, reresh this page

Coursera Learners:

- Coursera only remembers your highest score. If you re-attempt the lab and get a lower score, it will not be reflected in Coursera
- If your best grade does not get reflected in Coursera for more than 10 minutes, please let the system administrators know at labs@colfaxresearch.com