Saint Petersburg National Research University of Information Technologies, Mechanics and Optics (ITMO University)
Faculty of Informational Technologies and Programming

REPORT

about laboratory work № 2

« Definite integral calculation»

Student
Nerinovsky Arseny **_M41331_**
(Surname, initials) Group

Saint-Petersburg, 2018

Report

June 28, 2019

1 GOAL OF LABORATORY WORK

Using the trapezoidal rule of definite integral to approximate the regions or compartmentalized sections under a graph of a given function and also calculating the area as well.

2 TASK DEFINITION

The specific problem that is solved in the laboratory work (1 paragraph). Calculating the execution time of a serial program, that is, the estimated time required for calculating the values of different upper and lower bounds of an integral, which is given as follows:

Choosing an arbitrary value of epsilon, with a maximum value of 10^-7. Convert the serial program written to a parallel program using OpenMP by using reduction concept and synchronization methods, such as; lock, atomic and critical sections. Later, count the speedup of executions based on different thread numbers.

3 BRIEF THEORY

OpenMP (Open Multi-Processing) is an application programming interface (API) that supports multi-platform shared memory multiprocessing programming in C, C++, and Fortran on most platforms, instruction set architectures and operating systems, including Solaris, AIX, HP-UX, Linux, macOS, and Windows. It consists of a set of compiler directives, library routines, and environment variables that influence run-time behaviorAn application built with the hybrid model of parallel programming can run on a computer cluster using both OpenMP and Message Passing Interface (MPI), such that OpenMP is used for parallelism within a (multi-core) node while MPI is used for parallelism between nodes. There have also been efforts to run OpenMP on software distributed shared memory systems, to translate OpenMP into MPI and to extend OpenMP for non-shared memory

$$f(x) = \frac{1}{x^2}\sin^2\left(\frac{1}{x}\right), \quad 0 < A << 1$$

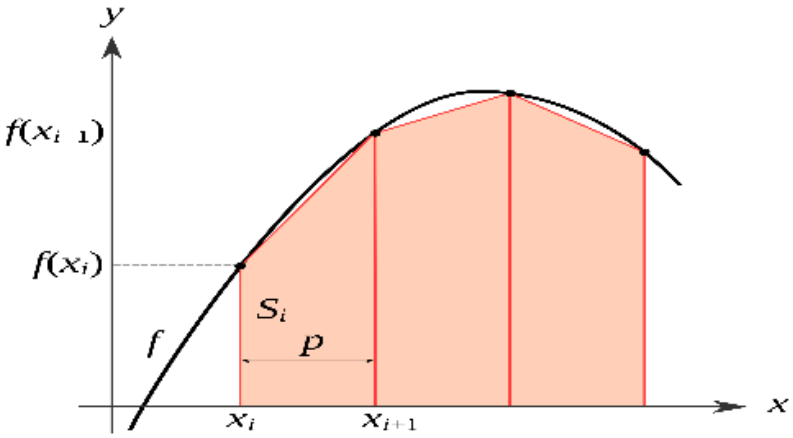
$$J(A,B) = \int_A^B \frac{1}{x^2}\sin^2\left(\frac{1}{x}\right)dx = -\frac{1}{2x} + \frac{1}{4}\sin\left(\frac{2}{x}\right)\Bigg|_A^B$$

systems. Synchronization Methods in OpenMP. To control issues pertaining to race conditions, synchronization methods are used to protect data conflicts.

- Reduction – reduction clause: reduction(op:list). A local copy of each list of variable is made and initialized depending on the op. Example; 1 for '*', the sign is assigned to number 1 as its identity. Updates occur on the local copy, then reduced into a single value and combined with the original global value. The construct for reduction is combined with the for construct: #pragma omp parallel for reduction(+:plus)
- Critical – allows on thread of code execute at a time. If a thread is performing a computation and hits a critical section in a block which isn't free, it has to wait for its turn. When a thread finishes its computation, it releases for the next thread in queue. The construct for critical section is #pragma omp critical; and it consumes all executions do by individual threads every time, i.e. on every overhead.
- Lock – is the lowest level of mutual exclusion synchronization; Lock Routines: omp_init_lock(), omp_set_lock(), omp_unset_lock(), omp_destroy_lock(), and omp_test_lock. Setting locks on essential section of codes in parallel for a thread at a time. Memory is released for the next thread when it has completed it task. Sometimes, with the help of omp_test_lock(), it queries to find free threads.
- Atomic – allows for quick updates of values in memory. It applies to a simple binary operation when updating a value. Example of its use is during an increment, decrement or performing operations such as read or write. The construct for atomic section is #pragma omp atomic.

3.1 Trapezoidal Rule

The trapezoidal rule is a technique for approximating the definite integral by approximating the region under the graph of the function as a trapezoid and calculating its area. Most often used in numerical analysis.



img

4 ALGORITHM (METHOD) OF IMPLEMENTATION

[28]:
```
%cat int.c
```

```
#include <omp.h>

#include <stdio.h>
#include <stdlib.h>


// #include <boost/math/quadrature/trapezoidal.hpp>

#define A ((double)_A)
#define B ((double)_B)
#define N ((double)_N)
#define PREC ((double)_PREC)


#ifndef INTEG
# define INTEG integrate0
#endif


#ifndef COMPUTE_SUM
# define COMPUTE_SUM compute_sum_reduction
#endif

#include <math.h>
```

```
double f(double x) {
        double a = (1./x) * sin(1./x);
        return pow(a, 2);
}

double f_int(double a, double b) {
        return (1./4.) * (  2. * (b-a)/(a*b) + sin(2/b) - sin(2/a) );
}

double F(double x) {
        return -1/(2*x) + (1/4) * sin(2/x);
}



double compute_sum_raw(size_t p, double h, double a){
        double sum = 0;

        // #pragma omp parallel for
        for(size_t j = 1; j < p; j += 1)
        {
                double y = f(a + j*h);
                sum += y;
        }
        return sum;
}

double compute_sum_atomic(size_t p, double h, double a){
        double sum = 0;

        #pragma omp parallel for
        for(size_t j = 1; j < p; j += 1)
        {
                double y = f(a + j*h);
                #pragma omp atomic
                sum += y;
        }
        return sum;
}

double compute_sum_critical(size_t p, double h, double a){
        double sum = 0;

        #pragma omp parallel for
        for(size_t j = 1; j < p; j += 1)
        {
                double y = f(a + j*h);
```

```
                #pragma omp critical
                sum += y;
        }
        return sum;
}

double compute_sum_reduction(size_t p, double h, double a){
        double sum = 0;

        #pragma omp parallel for reduction(+:sum)
        for(size_t j = 1; j < p; j += 1)
        {
                double y = f(a + j*h);
                sum += y;
        }
        return sum;
}

double compute_sum_locks(size_t p, double h, double a){
        double sum = 0;
        omp_lock_t lock;

        omp_init_lock(&lock);

        #pragma omp parallel for
        for(size_t j = 1; j < p ; j += 1)
        {
                double y = f(a + j*h);

                omp_set_lock(&lock);
                sum += y;
                omp_unset_lock(&lock);
        }
        omp_destroy_lock(&lock);

        return sum;
}


#define compute_sum compute_sum_atomic


double integrate0() {
        double a = A, b = B;
    double ya = f(a), yb = f(b);
    double h, error;
    double sum1 = 0, sum2 = 0, sum;
```

```
    int n = 4;

    for(size_t i = 0; i < 1e6; i += 50) {


        n += 1;
        h = (b - a)/n;

        sum = (ya + yb)*0.5;
        sum += COMPUTE_SUM(n, h, a);
        sum *= h;

                sum1 = sum2;
                sum2 = sum;
        if(i != 0) {
                error = fabs(sum2-sum1)/fabs(sum2);
                if(fabs(error) <= fabs(PREC)) {
                        break;
                }
        }
    }

    printf("points: %d\n", n);
    return sum2;
}

double integrate1() {
        double a = A, b = B;
    double ya = f(a), yb = f(b);
    double h, error;
    double sum1 = 0, sum_res = 0;
    double *sums, sum;
    int n = 4, end_work = 0;

    #pragma omp parallel shared(n, sums, sum, end_work, sum_res)
    {
        int tid = omp_get_thread_num();
        int n_threads = omp_get_num_threads();


        #pragma omp single
        {
                sums = (double*)malloc(n_threads * sizeof(double));
        }

        // #pragma omp barrier
```

```
        for(size_t i = 0; i < 1e6; ++i, n += n_threads) {
            int __end_work;
            int _n = n + tid;

            double h = (b - a)/_n;

            sums[tid] = (ya + yb)*0.5;

                    // #pragma omp for reduction(+:sum) nowait
                    #pragma omp taskloop nowait
                            for(size_t j = 1; j < _n; j += 1) {
                                    sums[tid] += f(a + j*h);
                            }
            sums[tid] *= h;


            #pragma omp barrier




            #pragma omp single
            {
                    for(size_t j = 1; j < n_threads; ++j) {
                            error = fabs(sums[j-1] - sums[j])/fabs(sums[j]);
                            // printf("%d %.9lf\n", tid, error);
                            if(fabs(error) <= fabs(PREC)) {
                                    // printf("DONE\n");

                                    #pragma omp atomic write
                                            end_work = 1;

                                    sum_res = sums[j];
                                    break;
                            }
                    }
                }
                // #pragma omp barrier


            #pragma omp atomic read
                            __end_work = end_work;
            if(__end_work) break;

        }

}
printf("points: %d\n", n);
```

```
    return sum_res;
}


// double integrate_boost(int max_refinements, double tol) {
//      double a = A, b = B;
//     double ya = f(a), yb = f(b);
//     double h = (b - a)*0.5;
//     double I0 = (ya + yb)*h;
//     double IL0 = (abs(ya) + abs(yb))*h;

//     double yh = f(a + h);
//     double I1;
//     I1 = I0*0.5 + yh*h;

//     // The recursion is:
//     // I_k = 1/2 I_{k-1} + 1/2^k \sum_{j=1; j odd, j < 2^k} f(a + j(b-a)/2^k)
//     size_t k = 2;
//     // We want to go through at least 4 levels so we have sampled the
function at least 10 times.
//     // Otherwise, we could terminate prematurely and miss essential features.
//     // This is of course possible anyway, but 10 samples seems to be a
reasonable compromise.
//     double error = abs(I0 - I1);
//     int points = 0;
//     while (k < 4 || (k < max_refinements && error > tol*abs(I1)) )
//     {
//         I0 = I1;

//         I1 = I0*0.5;
//         size_t p = static_cast<size_t>(1u) << k;
//         points += p/2;
//         h *= 0.5;

//         double sum = COMPUTE_SUM(h, p, a);

//         I1 += sum*h;
//         ++k;
//         error = abs(I0 - I1);
//     }
//     printf("points: %d\n", points);
//     return I1;
// }


int main(int argc, char *argv[])
{
```

```
        double sum = INTEG();

        // using boost::math::quadrature::trapezoidal;
        // double I = trapezoidal(f, A, B,
boost::math::tools::root_epsilon<double>(), 10);

        printf("res_prec: %f\n", f_int(A, B));
        printf("res_prec2: %f\n",F(B)-F(A));
        printf("res: %f\n", sum);
        // printf("res_boost: %f\n", I);
        // printf("precision: %f\n", fabs(sum - res_prec) / fabs(res_prec));

        return 0;
```

5 RESULT AND EXPERIMENTS

Task:

1. Choose precision
2. Calculate integral with different A and B values from the table
3. Calculate execution time of serial program
4. Write a parallel program with:
    1. atomic
    2. Critical sections
    3. Locks
    4. reduction
5. Count speedup with different thread number
6. Fill the table (for each point of 4a-4d)

[1]:
```python
import subprocess
import os


def compile(*defs, **defskw):
    args = [f"-D{k}" for k in defs] + [f"-D{k}={v}" for k, v in defskw.items()]
    _cmd = 'g++ int.c -o int -fopenmp -lm -I boost_1_70_0/ -std=c++14'.split()␣
 ↪+ args
    # print(' '.join(_cmd))
    cmd = subprocess.run(_cmd, stdout=subprocess.PIPE, stderr=subprocess.PIPE)

    if(cmd.stdout): print('cmd.stdout', cmd.stdout)
    if(cmd.stderr): print('cmd.stderr', cmd.stderr)

def run(env=None):
    cmd = subprocess.run('./int', stdout=subprocess.PIPE, stderr=subprocess.
 ↪PIPE, env=env)
```

```
    return cmd.stdout.decode('utf8')
```

5.1 Reduce configuraions

Lets test different reduce configuratios.

[2]:
```
env = os.environ.copy()
env['OMP_NUM_THREADS'] = str(4)

for x in ['raw', 'atomic', 'locks', 'reduction', 'critical']:
    print(f'Executing integration with {x} locking')
    compile(_A=1, _B=10, _PREC=-6.30E-11, COMPUTE_SUM=f'compute_sum_{x}')

    %timeit run(env)
    print()
```

```
Executing integration with raw locking
cmd.stderr b"clang: warning: treating 'c' input as 'c++' when in C++ mode, this
behavior is deprecated [-Wdeprecated]\nclang: error: unsupported option
'-fopenmp'\nclang: error: unsupported option '-fopenmp'\n"


      ␣
↪---------------------------------------------------------------------------

       OSError                                   Traceback (most recent call␣
↪last)

       <ipython-input-2-e7cd551b4f7f> in <module>
         6     compile(_A=1, _B=10, _PREC=-6.30E-11,␣
↪COMPUTE_SUM=f'compute_sum_{x}')
         7
   ----> 8     get_ipython().run_line_magic('timeit', 'run(env)')
         9     print()
        10


       /usr/local/lib/python3.7/site-packages/IPython/core/interactiveshell.py␣
↪in run_line_magic(self, magic_name, line, _stack_depth)
      2305                 kwargs['local_ns'] = sys._getframe(stack_depth).
↪f_locals
      2306             with self.builtin_trap:
   -> 2307                 result = fn(*args, **kwargs)
      2308             return result
      2309
```

```
      </usr/local/lib/python3.7/site-packages/decorator.py:decorator-gen-60>␣
↪in timeit(self, line, cell, local_ns)


      /usr/local/lib/python3.7/site-packages/IPython/core/magic.py in␣
↪<lambda>(f, *a, **k)
      185     # but it's overkill for just that one bit of state.
      186     def magic_deco(arg):
  --> 187         call = lambda f, *a, **k: f(*a, **k)
      188
      189         if callable(arg):


      /usr/local/lib/python3.7/site-packages/IPython/core/magics/execution.py␣
↪in timeit(self, line, cell, local_ns)
     1145             for index in range(0, 10):
     1146                 number = 10 ** index
  -> 1147                 time_number = timer.timeit(number)
     1148                 if time_number >= 0.2:
     1149                     break


      /usr/local/lib/python3.7/site-packages/IPython/core/magics/execution.py␣
↪in timeit(self, number)
      159         gc.disable()
      160         try:
  --> 161             timing = self.inner(it, self.timer)
      162         finally:
      163             if gcold:


      <magic-timeit> in inner(_it, _timer)


      <ipython-input-1-d1c53e595c3b> in run(env)
       13
       14 def run(env=None):
  ---> 15     cmd = subprocess.run('./int', stdout=subprocess.PIPE,␣
↪stderr=subprocess.PIPE, env=env)
       16     return cmd.stdout.decode('utf8')
       17


      /usr/local/Cellar/python/3.7.3/Frameworks/Python.framework/Versions/3.7/
↪lib/python3.7/subprocess.py in run(input, capture_output, timeout, check,␣
↪*popenargs, **kwargs)
      470         kwargs['stderr'] = PIPE
```

```
      471
--> 472     with Popen(*popenargs, **kwargs) as process:
      473         try:
      474             stdout, stderr = process.communicate(input,␣
↪timeout=timeout)


      /usr/local/Cellar/python/3.7.3/Frameworks/Python.framework/Versions/3.7/
↪lib/python3.7/subprocess.py in __init__(self, args, bufsize, executable,␣
↪stdin, stdout, stderr, preexec_fn, close_fds, shell, cwd, env,␣
↪universal_newlines, startupinfo, creationflags, restore_signals,␣
↪start_new_session, pass_fds, encoding, errors, text)
      773                                 c2pread, c2pwrite,
      774                                 errread, errwrite,
--> 775                                 restore_signals, start_new_session)
      776         except:
      777             # Cleanup if the child failed starting.


      /usr/local/Cellar/python/3.7.3/Frameworks/Python.framework/Versions/3.7/
↪lib/python3.7/subprocess.py in _execute_child(self, args, executable,␣
↪preexec_fn, close_fds, pass_fds, cwd, env, startupinfo, creationflags, shell,␣
↪p2cread, p2cwrite, c2pread, c2pwrite, errread, errwrite, restore_signals,␣
↪start_new_session)
     1520                         if errno_num == errno.ENOENT:
     1521                             err_msg += ': ' + repr(err_filename)
-> 1522                     raise child_exception_type(errno_num, err_msg,␣
↪err_filename)
     1523                 raise child_exception_type(err_msg)
     1524


      OSError: [Errno 8] Exec format error: './int'
```

We clearly see the advatange of using OpenMP reductions..

5.2 Thread speedup

Lets analyze how the number of threads affects the speed of the execution.

[3]:
```
env = os.environ.copy()
env['OMP_NUM_THREADS'] = str(4)


compile(_A=1, _B=10, _PREC=-6.30E-11, COMPUTE_SUM=f'compute_sum_reduction')

print(f'Executing integration with 4 threads')
```

```
%timeit run(env)

env = os.environ.copy()
env['OMP_NUM_THREADS'] = str(2)


compile(_A=1, _B=10, _PREC=-6.30E-11, COMPUTE_SUM=f'compute_sum_reduction')

print(f'Executing integration with 2 threads')
%timeit run(env)

env = os.environ.copy()
env['OMP_NUM_THREADS'] = str(1)

print(f'Executing integration with 1 threads (!!)')
compile(_A=1, _B=10, _PREC=-6.30E-11, COMPUTE_SUM=f'compute_sum_reduction')

%timeit run(env)
```

```
cmd.stderr b"clang: warning: treating 'c' input as 'c++' when in C++ mode, this
behavior is deprecated [-Wdeprecated]\nclang: error: unsupported option
'-fopenmp'\nclang: error: unsupported option '-fopenmp'\n"
Executing integration with 4 threads
```

```
      ␣
↪---------------------------------------------------------------------------

       OSError                                   Traceback (most recent call␣
↪last)

       <ipython-input-3-3884ee48e90e> in <module>
         6
         7 print(f'Executing integration with 4 threads')
   ----> 8 get_ipython().run_line_magic('timeit', 'run(env)')
         9
        10 env = os.environ.copy()


       /usr/local/lib/python3.7/site-packages/IPython/core/interactiveshell.py␣
↪in run_line_magic(self, magic_name, line, _stack_depth)
      2305                 kwargs['local_ns'] = sys._getframe(stack_depth).
↪f_locals
      2306             with self.builtin_trap:
   -> 2307                 result = fn(*args, **kwargs)
      2308             return result
      2309
```

```
      </usr/local/lib/python3.7/site-packages/decorator.py:decorator-gen-60>␣
↪in timeit(self, line, cell, local_ns)


      /usr/local/lib/python3.7/site-packages/IPython/core/magic.py in␣
↪<lambda>(f, *a, **k)
      185     # but it's overkill for just that one bit of state.
      186     def magic_deco(arg):
  --> 187         call = lambda f, *a, **k: f(*a, **k)
      188
      189         if callable(arg):


      /usr/local/lib/python3.7/site-packages/IPython/core/magics/execution.py␣
↪in timeit(self, line, cell, local_ns)
     1145             for index in range(0, 10):
     1146                 number = 10 ** index
  -> 1147                 time_number = timer.timeit(number)
     1148                 if time_number >= 0.2:
     1149                     break


      /usr/local/lib/python3.7/site-packages/IPython/core/magics/execution.py␣
↪in timeit(self, number)
      159         gc.disable()
      160         try:
  --> 161             timing = self.inner(it, self.timer)
      162         finally:
      163             if gcold:


      <magic-timeit> in inner(_it, _timer)


      <ipython-input-1-d1c53e595c3b> in run(env)
       13
       14 def run(env=None):
  ---> 15     cmd = subprocess.run('./int', stdout=subprocess.PIPE,␣
↪stderr=subprocess.PIPE, env=env)
       16     return cmd.stdout.decode('utf8')
       17
```

```
      /usr/local/Cellar/python/3.7.3/Frameworks/Python.framework/Versions/3.7/
↪lib/python3.7/subprocess.py in run(input, capture_output, timeout, check,␣
↪*popenargs, **kwargs)
      470         kwargs['stderr'] = PIPE
      471
  --> 472     with Popen(*popenargs, **kwargs) as process:
      473         try:
      474             stdout, stderr = process.communicate(input,␣
↪timeout=timeout)


      /usr/local/Cellar/python/3.7.3/Frameworks/Python.framework/Versions/3.7/
↪lib/python3.7/subprocess.py in __init__(self, args, bufsize, executable,␣
↪stdin, stdout, stderr, preexec_fn, close_fds, shell, cwd, env,␣
↪universal_newlines, startupinfo, creationflags, restore_signals,␣
↪start_new_session, pass_fds, encoding, errors, text)
      773                                 c2pread, c2pwrite,
      774                                 errread, errwrite,
  --> 775                                 restore_signals, start_new_session)
      776         except:
      777             # Cleanup if the child failed starting.


      /usr/local/Cellar/python/3.7.3/Frameworks/Python.framework/Versions/3.7/
↪lib/python3.7/subprocess.py in _execute_child(self, args, executable,␣
↪preexec_fn, close_fds, pass_fds, cwd, env, startupinfo, creationflags, shell,␣
↪p2cread, p2cwrite, c2pread, c2pwrite, errread, errwrite, restore_signals,␣
↪start_new_session)
     1520                         if errno_num == errno.ENOENT:
     1521                             err_msg += ': ' + repr(err_filename)
  -> 1522                     raise child_exception_type(errno_num, err_msg,␣
↪err_filename)
     1523                 raise child_exception_type(err_msg)
     1524


      OSError: [Errno 8] Exec format error: './int'
```

More threads equal to a faster execution

5.2.1 Alternative integration implementaion

The program offered several ways to impove beyond the ordinary sum. An alternative version was also developed. In order to increase performance a combination of OpenMP task and OpenMP normal multithreading was used. No because of the design no reductions were possible.

[4]:
```
for ts in [2, 4,8, 16, 32]:
    env = os.environ.copy()
    env['OMP_NUM_THREADS'] = str(ts)

    compile(_A=1, _B=10, _PREC=-6.30E-11, INTEG='integrate1')

    print(f"Alternative tasks based integration with {ts} threas")
    %timeit run(env)

    compile(_A=1, _B=10, _PREC=-6.30E-11, INTEG='integrate0')

    print(f"Default tasks based integration with {ts} threas")
    %timeit run(env)
    print()
```

```
cmd.stderr b"clang: warning: treating 'c' input as 'c++' when in C++ mode, this
behavior is deprecated [-Wdeprecated]\nclang: error: unsupported option
'-fopenmp'\nclang: error: unsupported option '-fopenmp'\n"
Alternative tasks based integration with 2 threas


      ␣
↪---------------------------------------------------------------------------

       OSError                                   Traceback (most recent call␣
↪last)

       <ipython-input-4-be3e0545f0e9> in <module>
         6
         7     print(f"Alternative tasks based integration with {ts} threas")
   ----> 8     get_ipython().run_line_magic('timeit', 'run(env)')
         9
        10     compile(_A=1, _B=10, _PREC=-6.30E-11, INTEG='integrate0')


       /usr/local/lib/python3.7/site-packages/IPython/core/interactiveshell.py␣
↪in run_line_magic(self, magic_name, line, _stack_depth)
      2305                 kwargs['local_ns'] = sys._getframe(stack_depth).
↪f_locals
      2306             with self.builtin_trap:
   -> 2307                 result = fn(*args, **kwargs)
      2308             return result
      2309


       </usr/local/lib/python3.7/site-packages/decorator.py:decorator-gen-60>␣
↪in timeit(self, line, cell, local_ns)
```

```
      /usr/local/lib/python3.7/site-packages/IPython/core/magic.py in␣
↪<lambda>(f, *a, **k)
      185     # but it's overkill for just that one bit of state.
      186     def magic_deco(arg):
  --> 187         call = lambda f, *a, **k: f(*a, **k)
      188
      189         if callable(arg):


      /usr/local/lib/python3.7/site-packages/IPython/core/magics/execution.py␣
↪in timeit(self, line, cell, local_ns)
     1145             for index in range(0, 10):
     1146                 number = 10 ** index
  -> 1147                 time_number = timer.timeit(number)
     1148                 if time_number >= 0.2:
     1149                     break


      /usr/local/lib/python3.7/site-packages/IPython/core/magics/execution.py␣
↪in timeit(self, number)
      159         gc.disable()
      160         try:
  --> 161             timing = self.inner(it, self.timer)
      162         finally:
      163             if gcold:


      <magic-timeit> in inner(_it, _timer)


      <ipython-input-1-d1c53e595c3b> in run(env)
       13
       14 def run(env=None):
  ---> 15     cmd = subprocess.run('./int', stdout=subprocess.PIPE,␣
↪stderr=subprocess.PIPE, env=env)
       16     return cmd.stdout.decode('utf8')
       17


      /usr/local/Cellar/python/3.7.3/Frameworks/Python.framework/Versions/3.7/
↪lib/python3.7/subprocess.py in run(input, capture_output, timeout, check,␣
↪*popenargs, **kwargs)
      470         kwargs['stderr'] = PIPE
      471
  --> 472     with Popen(*popenargs, **kwargs) as process:
```

```
      473         try:
      474             stdout, stderr = process.communicate(input,␣
↪timeout=timeout)


      /usr/local/Cellar/python/3.7.3/Frameworks/Python.framework/Versions/3.7/
↪lib/python3.7/subprocess.py in __init__(self, args, bufsize, executable,␣
↪stdin, stdout, stderr, preexec_fn, close_fds, shell, cwd, env,␣
↪universal_newlines, startupinfo, creationflags, restore_signals,␣
↪start_new_session, pass_fds, encoding, errors, text)
      773                                 c2pread, c2pwrite,
      774                                 errread, errwrite,
   --> 775                                 restore_signals, start_new_session)
      776         except:
      777             # Cleanup if the child failed starting.


      /usr/local/Cellar/python/3.7.3/Frameworks/Python.framework/Versions/3.7/
↪lib/python3.7/subprocess.py in _execute_child(self, args, executable,␣
↪preexec_fn, close_fds, pass_fds, cwd, env, startupinfo, creationflags, shell,␣
↪p2cread, p2cwrite, c2pread, c2pwrite, errread, errwrite, restore_signals,␣
↪start_new_session)
     1520                         if errno_num == errno.ENOENT:
     1521                             err_msg += ': ' + repr(err_filename)
   -> 1522                     raise child_exception_type(errno_num, err_msg,␣
↪err_filename)
     1523                 raise child_exception_type(err_msg)
     1524


      OSError: [Errno 8] Exec format error: './int'
```

We clearly see the advantage of the task based design which maximize resource utilization without compromizing flexiblity.

6 Filling the data

Lets collect the data required by the task

[5]:
```
xs = [
    ( 0.00001 , 0.0001 , -2.77e-11),
    (  0.0001 ,  0.001 ,   1.9e-10),
    (   0.001 ,   0.01 ,  2.05E-11),
    (    0.01 ,    0.1 , -2.22E-12),
    (     0.1 ,      1 ,  8.67E-11),
    (       1 ,     10 , -6.00E-11),
    (      10 ,    100 , -6.30E-11)
```

```
]
for a, b, eps in xs:
    compile(_A=a, _B=b, _PREC=eps, INTEG='integrate1')
    print(f"Intergating on ({a}, {b}) interval with {eps} precision")
    print("Time taken:", end=' ')
    %timeit run(env)
    print(run().split('\n')[0])
    print()
```

```
cmd.stderr b"clang: warning: treating 'c' input as 'c++' when in C++ mode, this
behavior is deprecated [-Wdeprecated]\nclang: error: unsupported option
'-fopenmp'\nclang: error: unsupported option '-fopenmp'\n"
Intergating on (1e-05, 0.0001) interval with -2.77e-11 precision
```

```
      ␣
↪---------------------------------------------------------------------------

       OSError                                   Traceback (most recent call␣
↪last)

       <ipython-input-5-f27bee3d4a18> in <module>
        12     print(f"Intergating on ({a}, {b}) interval with {eps} precision")
        13     print("Time taken:", end=' ')
   ---> 14     get_ipython().run_line_magic('timeit', 'run(env)')
        15     print(run().split('\n')[0])
        16     print()

       /usr/local/lib/python3.7/site-packages/IPython/core/interactiveshell.py␣
↪in run_line_magic(self, magic_name, line, _stack_depth)
      2305                 kwargs['local_ns'] = sys._getframe(stack_depth).
↪f_locals
      2306             with self.builtin_trap:
   -> 2307                 result = fn(*args, **kwargs)
      2308             return result
      2309

       </usr/local/lib/python3.7/site-packages/decorator.py:decorator-gen-60>␣
↪in timeit(self, line, cell, local_ns)

       /usr/local/lib/python3.7/site-packages/IPython/core/magic.py in␣
↪<lambda>(f, *a, **k)
       185     # but it's overkill for just that one bit of state.
       186     def magic_deco(arg):
```

```
   --> 187         call = lambda f, *a, **k: f(*a, **k)
       188
       189         if callable(arg):


       /usr/local/lib/python3.7/site-packages/IPython/core/magics/execution.py␣
↪in timeit(self, line, cell, local_ns)
      1145             for index in range(0, 10):
      1146                 number = 10 ** index
   -> 1147                 time_number = timer.timeit(number)
      1148                 if time_number >= 0.2:
      1149                     break


       /usr/local/lib/python3.7/site-packages/IPython/core/magics/execution.py␣
↪in timeit(self, number)
       159         gc.disable()
       160         try:
   --> 161             timing = self.inner(it, self.timer)
       162         finally:
       163             if gcold:


       <magic-timeit> in inner(_it, _timer)


       <ipython-input-1-d1c53e595c3b> in run(env)
        13
        14 def run(env=None):
   ---> 15     cmd = subprocess.run('./int', stdout=subprocess.PIPE,␣
↪stderr=subprocess.PIPE, env=env)
        16     return cmd.stdout.decode('utf8')
        17


       /usr/local/Cellar/python/3.7.3/Frameworks/Python.framework/Versions/3.7/
↪lib/python3.7/subprocess.py in run(input, capture_output, timeout, check,␣
↪*popenargs, **kwargs)
       470         kwargs['stderr'] = PIPE
       471
   --> 472     with Popen(*popenargs, **kwargs) as process:
       473         try:
       474             stdout, stderr = process.communicate(input,␣
↪timeout=timeout)
```

```
      /usr/local/Cellar/python/3.7.3/Frameworks/Python.framework/Versions/3.7/
↪lib/python3.7/subprocess.py in __init__(self, args, bufsize, executable,␣
↪stdin, stdout, stderr, preexec_fn, close_fds, shell, cwd, env,␣
↪universal_newlines, startupinfo, creationflags, restore_signals,␣
↪start_new_session, pass_fds, encoding, errors, text)
      773                                 c2pread, c2pwrite,
      774                                 errread, errwrite,
   --> 775                                 restore_signals, start_new_session)
      776         except:
      777             # Cleanup if the child failed starting.


      /usr/local/Cellar/python/3.7.3/Frameworks/Python.framework/Versions/3.7/
↪lib/python3.7/subprocess.py in _execute_child(self, args, executable,␣
↪preexec_fn, close_fds, pass_fds, cwd, env, startupinfo, creationflags, shell,␣
↪p2cread, p2cwrite, c2pread, c2pwrite, errread, errwrite, restore_signals,␣
↪start_new_session)
     1520                         if errno_num == errno.ENOENT:
     1521                             err_msg += ': ' + repr(err_filename)
   -> 1522                     raise child_exception_type(errno_num, err_msg,␣
↪err_filename)
     1523                 raise child_exception_type(err_msg)
     1524


      OSError: [Errno 8] Exec format error: './int'
```

7 CONCLUSION

The three method (locks, atomics, critical sections) shows less performance because of spending a lot of time on synchronizing and blocks, while the operation of summing is very expensive. So most of time threads just waiting for another threads unlock (or store the new value in memory or send the sync signals). The reduction shows good boost up.