Numerical Methods; October–November 2023
Assignment 1
Due: Thursday, 16 November 2023

Comparing computers

The computing power of a computer is often measured in terms of floating-point operations per second or Flop/s[1] and is called the *theoretical peak performance*[2] of that computer. Consider a computer with an Intel Xeon Gold 6433NE processor. (This processor is also informally called the "Sapphire Rapids" processor. It is one of the Xeon series of processors manufactured by Intel. Xeon processors are seldom found on personal laptops or desktops, but they are quite common in supercomputers.) Intel's documentation[3] tells us that this processor has 32 cores, each running with a clock frequency of 2 GHz. If each core can do 16 floating-point operations per clock cycle[4], then the theoretical peak performance of this computer is floating point operations per clock cycle × number of clock cycles per second × number of cores, which is

$$16 \times 2\text{ billion} \times 32 = 1024\,\text{billion}\,\text{Flop/s} = 1\,\text{TFlop/s}. \tag{1}$$

► Now that you know how to estimate the theoretical peak performance of a computer, what is the theoretical peak performance of a high-performance-computing (HPC) cluster with 32 connected nodes, with each node having *two* Intel Xeon Gold 6433NE processors?

► TIFR's Computer Center and Communication Facility (CCCF) has an HPC cluster with 20 compute nodes. Consult the documentation on CCCF's web site to find out what processors these nodes have. Then estimate the theoretical peak performance of these 20 nodes.

► What CPU does your personal laptop or desktop have? Estimate its theoretical peak performance.

We discussed in the class that GPUs are a kind of processor that can sometime be much better suited for our work than CPUs. Figuring out the theoretical peak performance of GPUs is easiest done by simply asking the GPU manufacturer. Consider the Hopper H100 SXM GPU made by the American company Nvidia. Nvidia's documentation[5] tells us that the theoretical peak performance of this GPU is 67 TFlop/s.

[1] A 'floating-point operation' is a mathematical operation involving real numbers, as opposed to integers. "Flop/s" is also sometimes written as "flops".

[2] When you actually run a code involving real numbers on this computer, the performance will usually be degraded due to various other factors.

[3] `https://ark.intel.com/content/www/us/en/ark/products/236590/intel-xeon-gold-6433ne-processor-60m-cache-2-00-ghz.html`

[4] It can be difficult to figure this out from Intel's documentation. A handy web site for such details is WikiChip. See, for example, `https://en.wikichip.org/wiki/flops` for information on floating-point-operations per cycle for various processors manufactured by Intel.

[5] `https://resources.nvidia.com/en-us-tensor-core/nvidia-tensor-core-gpu-datasheet`

▶ What is the theoretical peak performance of an HPC cluster with 16 connected GPUs? Is this cluster superior to the 32-node Xeon CPU cluster considered above? Suppose you have both of these clusters available. List the factors that will determine which cluster to use for a given physics problem.

TOP500[6] is a list of the best 500 computers in the world. This list is compiled by Erich Strohmaier (LBNL), Jack Dongarra (University of Tennessee), Horst Simon (LBNL), and Martin Meuer (Prometeus), and is published twice every year (in June and November).

▶ How many Indian computers are there in the June 2023 TOP500 list? How many Chinese computers are there? How many American computers are there?

▶ What is the theoretical peak performance[7] of the top-ranked computer in the June 2023 TOP500 list? What is the theoretical peak performance of the top-ranked Indian computer in this list?

▶ Sometimes, the physics problem at hand is memory-limited rather than compute-power-limited. Consider our 32-node Xeon CPU cluster (which we discussed above) again. Suppose the system administrator tells us that in this HPC cluster, the available memory per core is 12 GB. If my physics problem requires me to manipulate an array of numbers that takes 12.5 TB memory, will I be able to use this HPC cluster to solve this problem? What if the physics problem required an array that takes 12.1 TB memory? What if it took 10 TB memory?

▶ How much memory does your personal laptop or desktop have?

Python

Set up Python on your personal laptop or desktop. Or get an account on the CCCF machines and run Python there.

▶ Spend at least 15 minutes browsing through the Python library and language manuals. Get a feel for how information is arranged. Figure out how to find the version of Python you have on your computer.

▶ Run the Python interpreter on your computer's terminal, and at the `>>>` prompt, type `"Hello World!"`, including the quotation marks. What happens?

▶ Now open a text editor and create a file named `module1.py`. In this file write `print('Hello World!')`. This is a simple but correct and complete Python code. Run this code in two ways: (a) on your terminal by typing something like `python module1.py`; and (b) in a Python interpreter by typing `import module1`. Which way is easier?

▶ What happens if you type `1/0` in the Python interpreter? Write a Python code that divides

[6] `https://www.top500.org`

[7] Just read these numbers off the list.

1 by 0, 1, 2, 3, 4, and 5, respectively, and prints the result on the terminal. Use loops. When you divide by 0, the code should print the string `indeterminate` on the terminal. Use exception handling.

► Type these instructions one after another in a Python interpreter and understand the results:

```
2 ** 16
2 / 5
2 // 5
2 / 5.0

'delhi' + 'mumbai'
S = 'mumbai'
'navi ' + S
S * 5
S[:0]
'Coastal {0} and landlocked {1}.'.format('Mumbai', 'Delhi')

('x',)[0]
('x', 'y')[1]

L = [1,2,3] + [4,5,6]
L
L[:]
L[:0]
L[-2]
L[-2:]
([1,2,3] + [4,5,6])[2:4]
[L[2], L[3]]
L.reverse()
L
L.sort(); L
L.index(4)

{'a':1, 'b':2}['b']
D = {'x':1, 'y':2, 'z':3 }
D['w'] = 0
D['x'] + D['w']
D[(1,2,3)] = 4
list(D.keys()), list(D.values()), (1,2,3) in D

[[]], ["",[],(),{},None]

x = [2,3]
```

```
x[1] = 4
x = (2,3)
x[1] = 4
```

▶ Write a Python code in a single file to print the first ten Fibonacci numbers. Create a Git repository and add this file to it. Then turn the file into a module that has a function that returns n Fibonacci numbers, where n is the function argument. Write another file that imports this module and uses the function in it to print the first 100 Fibonacci numbers. Put this file into your Git repository. Publish your Git repository on Github and send us the link. Python comes with a module called `timeit`. Use this module and report how long it takes for your code to produce 100 Fibonacci numbers. Can you write a single-line Python code to print the first 10 Fibonacci numbers. (Hint: Use `lambda`.)

▶ Consider the string `S='mumbai'`. Write a Python code with a `for` loop that prints each character in this string and its Unicode code point. Unicode code points are integers and can be obtained using the `ord` built-in function. Next, change your loop to compute the sum of these Unicode code points of the characters in `S`. Now change your code again to return a list that contains the Unicode code points of the characters in `S`. Do this in three ways: by using list methods, by using list comprehension, and by using the `map` class.

▶ Consider the following Python code. It searches a list of numbers for 2^5.

```
L = [1, 2, 4, 8, 16, 32, 64]
X = 5

found = False
i = 0
while not found and i < len(L):
   if 2 ** X == L[i]:
      found = True
   else:
      i = i+1

if found:
   print('at index', i)
else:
   print(X, 'not found')
```

First, type this code in a file named `power.py`, add it to a Git repository, publish it on Github, and send us a Github link. Next, eliminate the `found` flag and the final `if` statement by using the Python's while-else construction. Now use Python's for-else construction and list `index` method to eliminate the explicit list-indexing logic. Next, remove the loop completely by using the `in` operator membership expression. Finally, use a `for` loop and the list *append* method to generate the powers-of-2 list `L` instead of hard-coding it. Do you think it would improve code performance to move the `2 ** X` expression outside the loops? How would you code that? (Commit each stage of this code development in Git.)

▶ In class, we wrote a function that adds two numbers. Write such a function in a file. Name the function `adder`. Now at the bottom of the file, call the function to add two strings, two lists, and two floating points. Run this code. Do you need a `print` statement to see the results? Now generalise the `adder` function to add an arbitrary number of arguments. Next, change the function so that it takes in three keyword arguments called `good`, `bad`, and `ugly`, with default values 1, 2, and 3 respectively. What happens if you now call `adder(ugly=1, good=5)`. Finally, generalise the function so that it can take an arbitrary numbers of keyword arguments.

Numpy, Scipy, and Matplotlib

▶ Make a plot of the spectrum of a black body at temperature 2.725 K.

The Cosmic Background Explorer (COBE) was an American space telescope that operated from 1989 to 1993. One of the three instruments on COBE was the Far-Infrared Absolute Spectrophotometer (FIRAS), which was used to measure the spectrum of the cosmic microwave background (CMB). The data measured by FIRAS is available from NASA[8] in a text file. Download the text file and read the first column into a Numpy array named `frequency`. Read the second column into a Numpy array named `cmb_flux`. (Use the `loadtxt` function from Numpy for reading the text file.) Now plot the CMB flux from the second array as a function of the frequency. Superimpose the curve on top of the black body spectrum in the previous plot. Pay attention to the units and discover that the two curves agree very well. This agreement, first measured using COBE/FIRAS, got the 2006 Nobel Prize. It showed that the CMB, which was formed in the universe more than 13 billion years ago, has a black body spectrum with a temperature of 2.725 K. Add axis labels and a legend to your plot to make it understandable. Save the plot to a PDF file and submit a printed copy.

There is a function in Numpy that you can use to find the wavelength or frequency at which the above black body spectra peak. Find this function and use it to find this wavelength. What part of the electromagnetic spectrum does this wavelength lie?

C

▶ Spend at least 15 minutes browsing through the C standard library documentation and GNU Scientific Library (GSL) documentation. Understand how the documentation is structured. Spot the API.

▶ Install the `gcc` C compiler on your computer. Using the `gcc` documentation, learn how to find out the version of the `gcc` compiler version installed. What version of this compiler do you have? When was this version published?

▶ Write a C code that prints the size of the `char`, `int`, `long int`, `float`, `double` data types on your computer in bytes.

[8] `https://lambda.gsfc.nasa.gov/data/cobe/firas/monopole_spec/firas_monopole_spec_v1.txt`

▶ Write a C code that prints the maximum and minimum machine numbers available on your computer for the `char`, `int`, `long int`, `float`, `double` data types.

▶ Write a C code to do the following:

1. Using the `malloc` standard library function, create an array of hundred `float` elements.
2. Assign the values 1^2, 2^2, …, 100^2 to the elements of this array.
3. Create a function that takes in an array of arbitrary length and returns an array of two elements. The first element of the output array is the mean of the input array. The second element of the output array is the variance of the input array. Use loops to calculate the mean and variance.
4. Using this function, calculate the mean and variance of the array you constructed in step 2 and print these to your terminal.
5. Free the memory used for your array.

Now modify the above code by moving your function to a second file. Use a header file to declare the function prototype. Use a Makefile to compile the three files over which your code is now spread.

Again modify your code. This time, while assigning values to the array, print the values of the elements of the array on the terminal *before* the assignment.

Now repeat the above step (that is, print the array element values before assigning the squares to them) but use `calloc` instead of `malloc`. What changed?

Now make another modification to the code. Write the result of your computation to a text file instead of the terminal. After running the code, open the text file using your text editor and confirm that it contains the two numbers that you expect it to contain.

Next, instead of writing the result of your computation to a text file, write it to a binary file. Open the binary file in your text editor. What do you see? Now write a second C code to read the contents of this binary file.

Using the `du` command on GNU/Linux, find out the file sizes of the text file and the binary files that you created above? Which one's smaller? Which file format will you use to store large amount of data?

Finally, change your code one last time to use GSL functions to compute the mean and variance instead of your loops. Print the result of both computational methods on the terminal. Are they identical?

(Each of the above modification should be a Git commit in your repository.)

C versus Python

▶ Write code to compute the factorial of a given integer n using four different techniques:

1. Using C

2. Using standard Python
3. Using Python, but with the `factorial` function from SciPy
4. Using Python, but writing the factorial function in C

In each case, you should have a function that computes the factorial of an arbitrary integer (use recursion) and then a main program that invokes this function to compute the factorial of a particular integer.

Now, compute the factorial of some large number and report the time taken to do this in each of the above four codes. (Use suitable C/Python library functions to measure time while the code is running.) Which technique is the fastest?