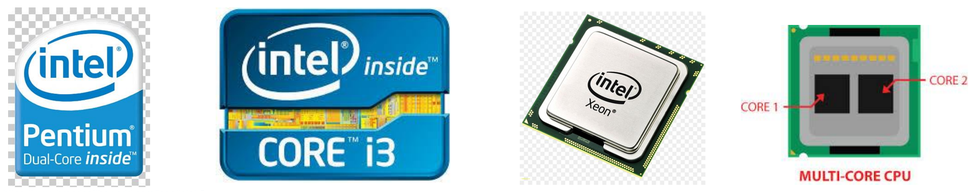
Multiprocessing in Python to Exploit Multicore Hardware

Rahul Roychowdhury, Dr. Eric Aubanel, Dr. Suprio Ray
University of New Brunswick, Faculty of Computer Science
rroychow@unb.ca, aubanel@unb.ca, sray@unb.ca
Charlie Gracie
IBM Canada
Charlie_Gracie@ca.ibm.com

Motivation

- Nowadays multi-core CPUs have become a standard in modern computer architecture. Multi-core CPUs are found not only in supercomputers, but also in personal laptops and mobiles.

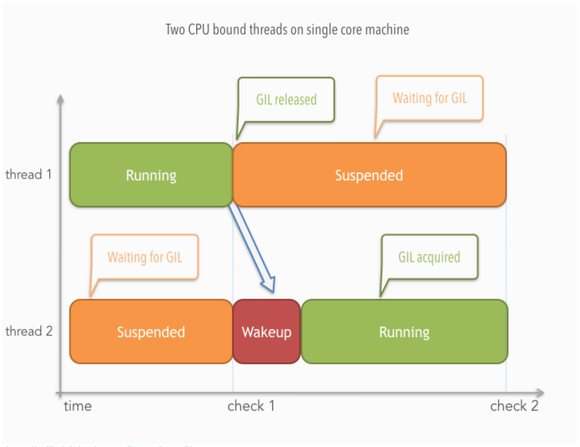


https://www.brandsoftheworld.com/logo/intel-6

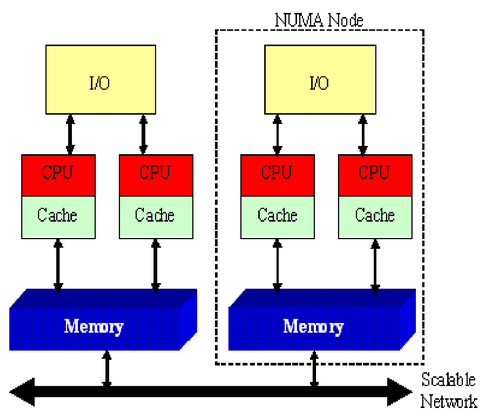
- This has given tremendous computing power to such devices.

On the other hand **Python** is becoming a very popular language for software development and data analysis but Global Interpreter Lock or **GIL** ensures that only one thread runs in the interpreter at once.

- It executes statements serially.
- This property of Python prohibits parallel processing and the power of modern multi-core CPUs cannot be utilized.



https://callhub.io/understanding-python-gil/



https://blog.e-zest.com/non-uniform-memory-architecture-numa/

- To overcome the disadvantages of GIL, Python utilizes the multiprocessing module and other frameworks, which spawns multiple sub-processes and allows to execute Python code in parallel.

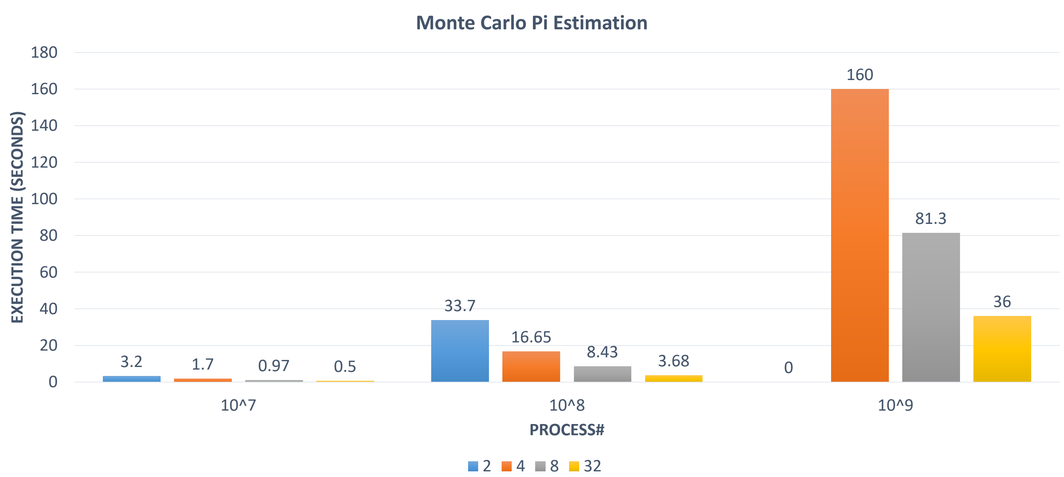
Problem Statement

- Research will be specifically conducted into enabling parallelism while dealing with Python's Global Interpreter Lock (GIL) and utilize the advantages of NUMA architecture.

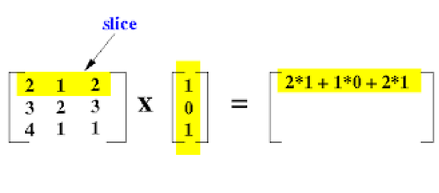
Experiments

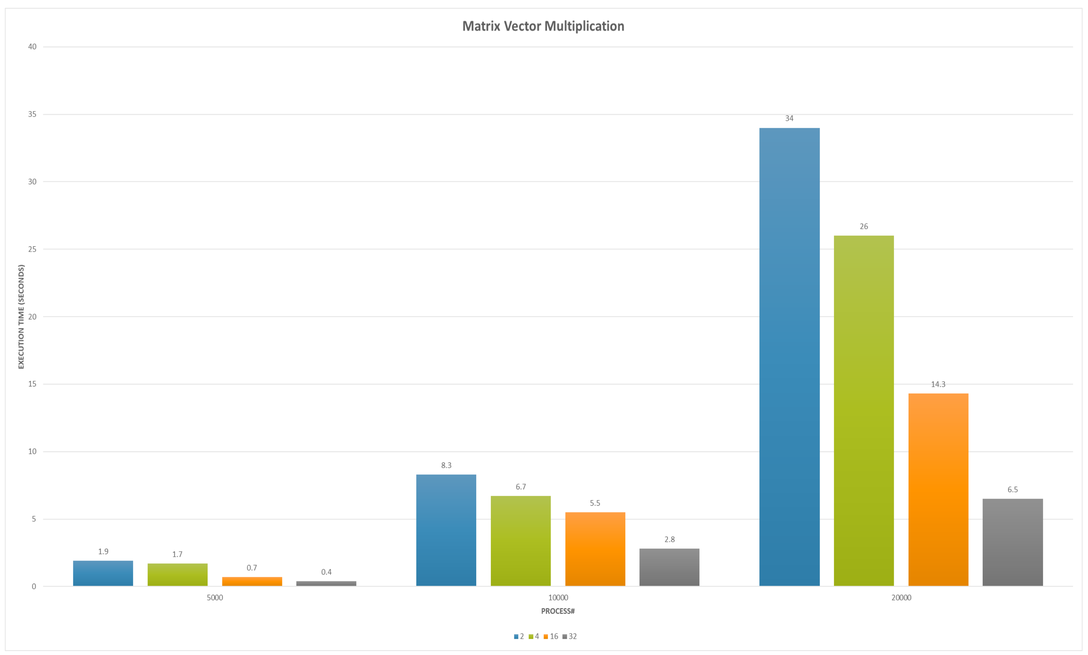
The following problem statements were executed in Python version 3.x and parallelized using multiprocessing module:

- Estimating the value of Pi using Monte Carlo method by generating a large number of random points and checking how many fall in the circle enclosed by the unit square.
- Each worker process gets *n* number of points to calculate Pi



- Parallel Matrix vector Multiplication





- **System configuration** – Intel Core i7, 32.0 GB(RAM),x64-based Processor, Linux(OS),32(CPU),4(NUMA node)

Future Work

- Explore various Python-based frameworks, which supports parallel processing
- NUMA-aware parallel processing using Python