OpenMP

Tutorial Session

OpenMP

- OpenMP is an Application Program Interface (API) that may be used to explicitly direct *multi-threaded, shared memory parallelism*

- It is not intrusive on the original serial code in that the OpenMP instructions are made in programs interpreted by the compiler.

- OpenMP uses the fork-join model of parallel execution. All OpenMP programs begin with a single master thread which executes sequentially until a parallel region is encountered, when it creates a team of parallel threads (FORK).

- When the team threads complete the parallel region, they synchronize and terminate, leaving only the master thread that executes sequentially (JOIN).

PROs

- Prevalence of multi-core computers
- Requires less code modification than using MPI
- OpenMP directives can be treated as comments if OpenMP is not available
- Directives can be added incrementally

CONs

- OpenMP codes cannot be run on distributed memory computers (exception is Intel's OpenMP)
- Requires a compiler that supports OpenMP (most do)
- limited by the number of processors available on a single computer
- often have lower parallel efficiency
    - rely more on parallelizable loops
    - tend to have a higher % of serial code
    - Amdahl's Law - if 50% of code is serial, it will only half the wall clock time no matter how many processors

OpenMP in Data Science

- Various Python packages such as Numpy, Scipy and pandas can utilize OpenMP to run on multiple CPUs.

- OpenMP has a lot of functionality that can help you with more complex cases. For example, asynchronous tasks, explicit barrier and tasks synchronization points, blocks of code for execution in individual threads, more intricate controls for memory consistency and for partial loop serialization.

- With OpenMP, the threads number is reduced to one at each machine, messages volume in MPI is reduced, and the time/speed is improved greatly.

- Both task parallelism and data parallelism can be achieved using OpenMP in this way. The runtime environment allocates threads to processors depending on usage, machine load and other factors. The runtime environment can assign the number of threads based on environment variables, or the code can do so using functions.

Outline of Assignment (OpenMP)

Assignment (OpenMP) will cover following topics:

- understand the parallelization in python
- understand multiprocessing in Python using OpenMP

Outlook

- Data researchers need to
  - execute parallel programming to tackle large scale data or to achieve memory efficiency
  - have sound information in parallel programming for a strategic advantage in the data science area.