BST 234 - Lab IV Complexity Classes and Parallelization

Eleni Elia

2/14/18

Complexity Class Definitions

- (P):Polynomial time algorithms
  - A decision which may be solved using a deterministic Turing machine in a polynomial amount of time.

  *Deterministic Turing machine: A formal mathematical model for defining an abstract machine capable of simulating any algorithm. Don't worry about knowing this!
- (NP):Nondeterministic Polynomial time algorithms
  - Verifying an instance takes polynomial time.

  *Important to note this does not mean non-polynomial but instead means non-deterministic polynomial!
- (NPH):NP-Hard algorithms Problems that are at least as hard as the hardest problems in NP.

  Consider an NP-hard algorithm A that can call an oracle to solve A, then it solves B, an NP algorithm, in polynomial time.

Complexity Class Definitions continued

- (NPC):NP-Complete algorithms
  - Both NP and NP-Hard. Although the verification may be performed in polynomial time,there is no efficient way to locate a solution in the first place.
- (NC):Nick?s Class
  - Algorithms solvable in logarithmic time with a polynomial number of parallel processors.

  Intuitively, the logarithmic problem size on a polynomial number processors guarantees an efficient (interesting) speed up.

NC is a subset of P but whether NC=P is unproven, i.e. there may exist purely sequential algorithms that may not be significantly sped up.

Complexity Classes Relationship

Efficiency and Speedup

For a sequential algorithm A, the run-time complexity is $T_A(n)$.
Parallelized on p processors, the run-time complexity is $T_p(n)$.

Algorithm efficiency:

$$E_p(n) = \frac{T_1(n)}{pxT_p(n)}$$

Algorithm speed-up:

$$S_p(n) = \frac{T_A(n)}{T_p(n)}$$

Threading versus Multiprocessing

Various parallel architectures were introduced in the lecture notes. Notably, we introduced the parallel random access memory(PRAM) architecture. While useful in theory, real hardware is different and we achieve parallelization through two main concepts:

- Threading
- Multiprocessing

Threading versus Multiprocessing

Threading:

A single shared memory is accessible to individual treads/sequences
-Multiple people adding pieces to the same puzzle

Multiprocessing:

Each process is given its own memory space
-Each person working on their own part of the puzzle

Techniques for Parallelization

- Embarrassingly Parallel
- Divide-and-Conquer Strategies
- Pipelined Computations

Parallelization in Python

A few options for parallelizing code

- xmlrpclib
- threading
- multiprocessing
- pp (parallel python)

- Easy enough to augment existing code by dropping in the parallelization
- Important to note that each python interpreter has a single global interpreter lock (GIL)

Global Interpreter Lock - a thread-safe mechanism

- Executes only one statement at a time (so-called serial processing, or single-threading) to prevent conflicts between threads.
- We can spawn multiple subprocesses to avoid some of the GIL?s disadvantages.

https://www.youtube.com/watch?v=ph374fJqFPE
http://www.dabeaz.com/GIL/gilvis/fourthread.html

Vectorizing

Similar to R, see the numpy module for vectorize

References

https://pymotw.com/3/multiprocessing/index.html
https://pymotw.com/3/threading/index.html
http://www.davekuhlman.org/python_multiprocessing_01.html
http://www.dabeaz.com/GIL/gilvis/index.html