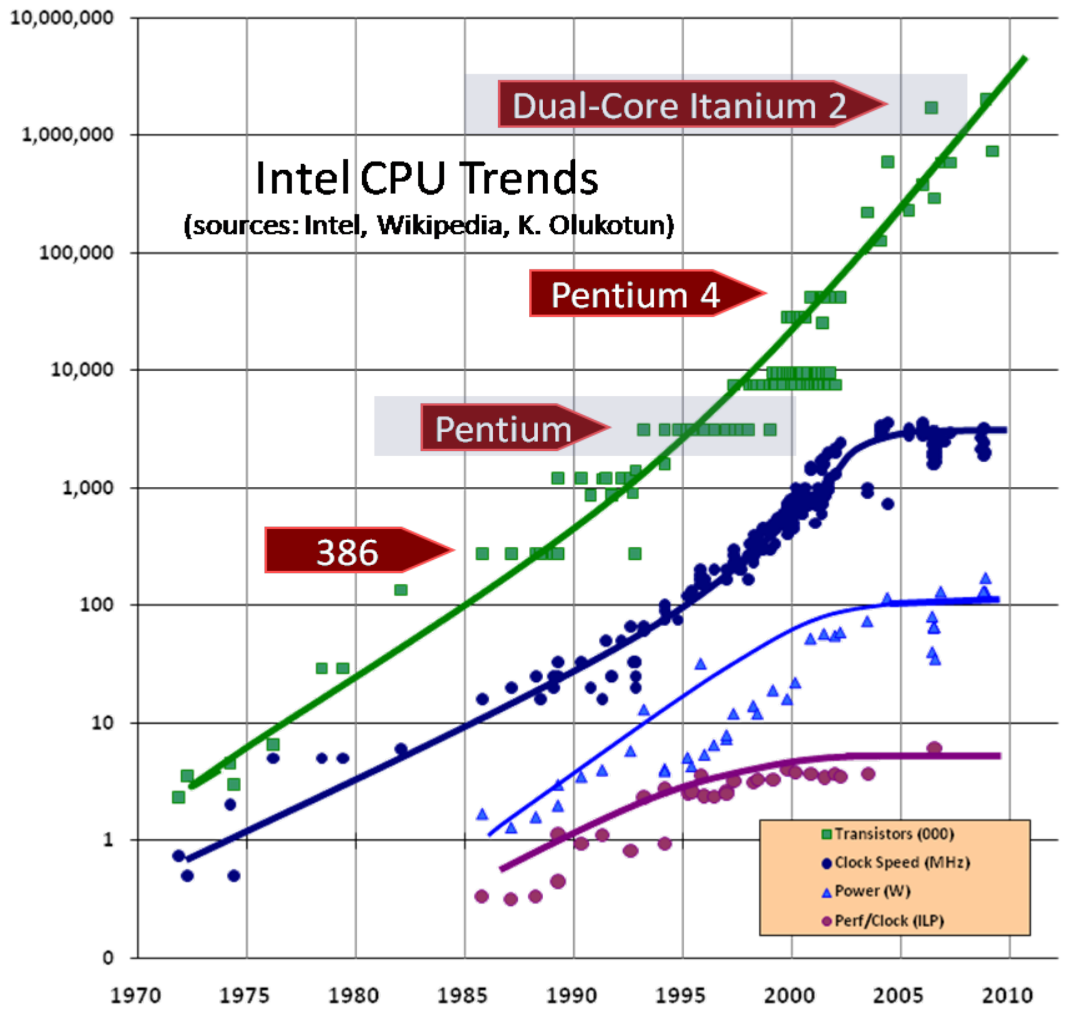
Multi-core programming with Python

Keerthi Shankar

The free lunch is over!

http://www.libertylive.org/

CPU Trends



http://www.gotw.ca/publications/concurrency-ddj.htm

How many cores do you have?

Entry-level Desktop Computers: 2 CPU cores

Mid-range Desktop Computers: 4 CPU cores

High-end Workstation Computers: 32 CPU cores

Upcoming Mobile Processors : 2 CPU cores!

What would you do as a Python programmer to catch up with this trend?

Evolution of threading in Python

thread - Multiple threads of control - Jurassic Age of Threading

threading - Higher level threading interface - A spiced up Jurassic Age (Has some sexier heroines like events, semaphores, etc)

multiprocessing - Process based "threading" interface - Resident Evil: After Life 3D!

Global Interpreter Lock

The Python interpreter is not fully thread safe. In order to support multi-threaded Python programs, there’s a global lock, called the global interpreter lock or GIL, that must be held by the current thread before it can safely access Python objects. Without the lock, even the simplest operations could cause problems in a multi-threaded program: for example, when two threads simultaneously increment the reference count of the same object, the reference count could end up being incremented only once instead of twice.

Does GIL hurt badly?

"Nevertheless, you're right the GIL is not as bad as you would initially think: you just have to undo the brainwashing you got from Windows and Java proponents who seem to consider threads as the only way to approach concurrent activities."

- Guido van Rossum

A demo using multiprocessing module

sys.exit(0)

keerthi.shankr@gmail.com