Shenzhen Winter Camp

Lecture 1

John Stachurski

2018

QuantEcon

Getting Started

These slides: `John/lecture1.pdf`

Warm up discussion: writing good code

- See notebook `John/supply_and_demand.ipynb`

See also lecture `Writing Good Code` on QuantEcon

QuantEcon

Programming Background — Software

A common classification:

- **low** level languages (assembly, C, Fortran)
- **high** level languages (Python, Ruby, Haskell)

Low level languages give us fine grained control

QuantEcon

Example. $1 + 1$ in assembly

```
pushq   %rbp
movq    %rsp, %rbp
movl    $1, -12(%rbp)
movl    $1, -8(%rbp)
movl    -12(%rbp), %edx
movl    -8(%rbp), %eax
addl    %edx, %eax
movl    %eax, -4(%rbp)
movl    -4(%rbp), %eax
popq    %rbp
```

QuantEcon

High level languages give us abstraction, automation, etc.

Example. Reading from a file in Python

```
data_file = open("data.txt")
for line in data_file:
    print(line.capitalize())
data_file.close()
```

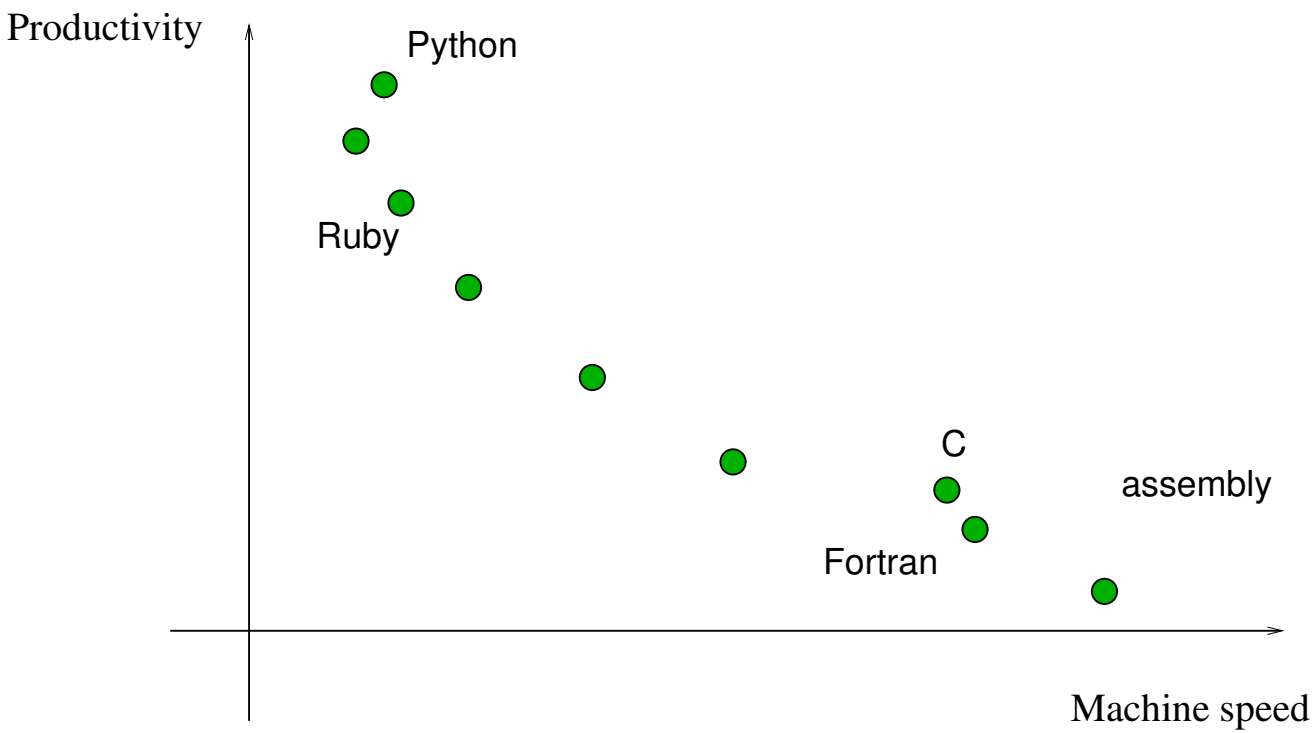
QuantEcon

Jane Street on readability:

> *There is no faster way for a trading firm to destroy itself than to deploy a piece of trading software that makes a bad decision over and over in a tight loop.*
>
> *Part of Jane Street's reaction to these technological risks was to put a very strong focus on building software that was easily understood—software that was readable.*
>
> – Yaron Minsky, Jane Street

QuantEcon

Trade-Offs



QuantEcon

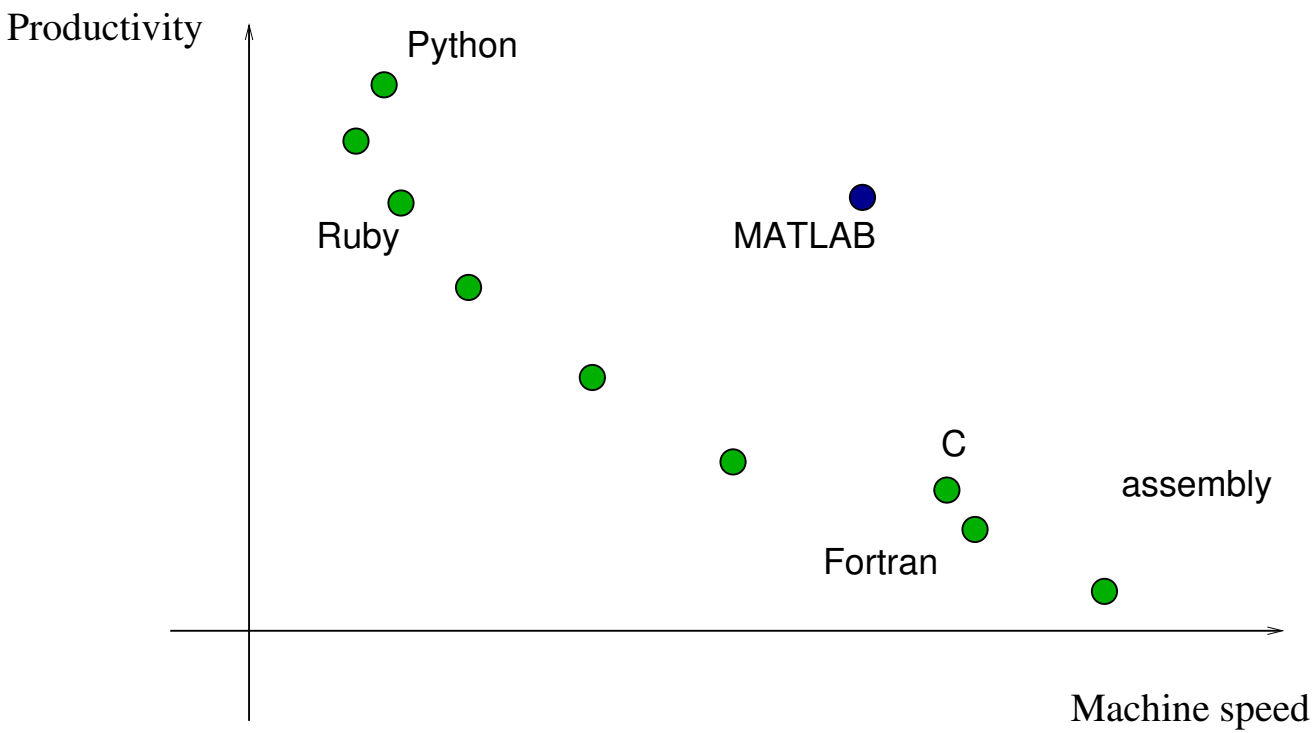
But what about scientific computing?

Requirements

- Productive — easy to read, write, debug, explore
- Fast computations

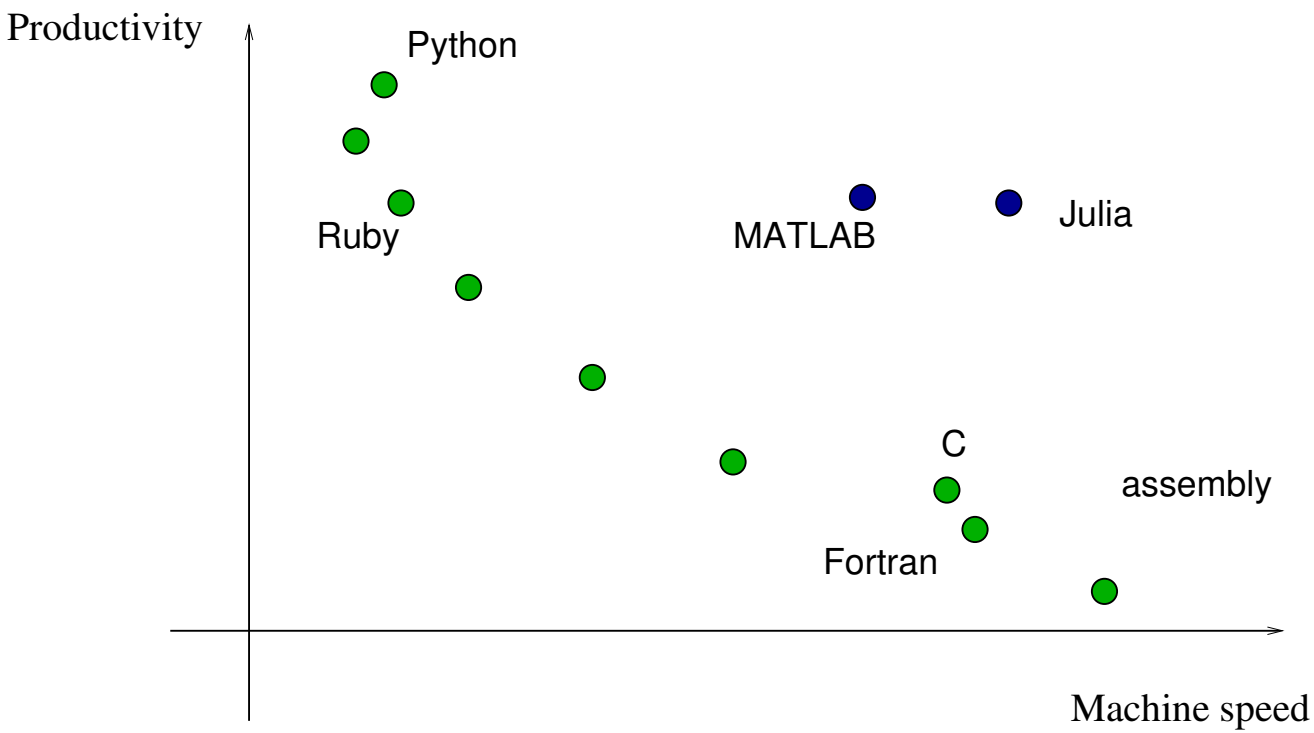
QuantEcon

Trade-Offs



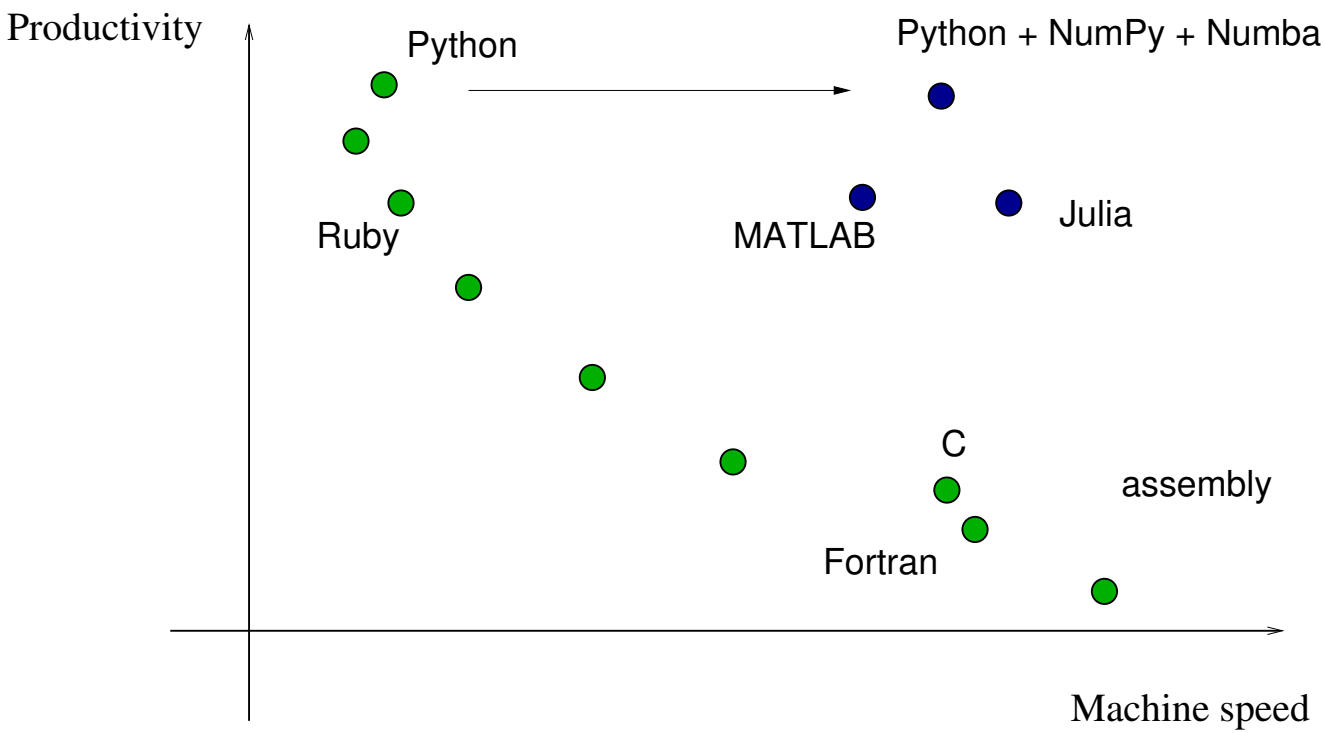
QuantEcon

Trade-Offs



QuantEcon

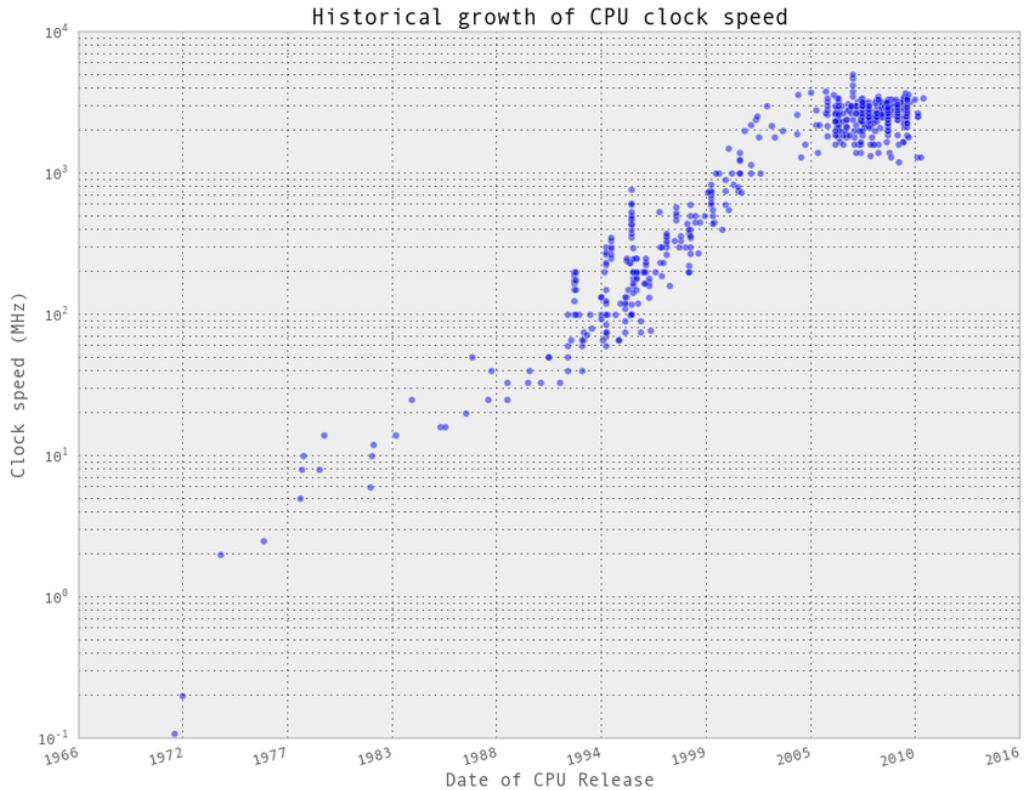
Trade-Offs



QuantEcon

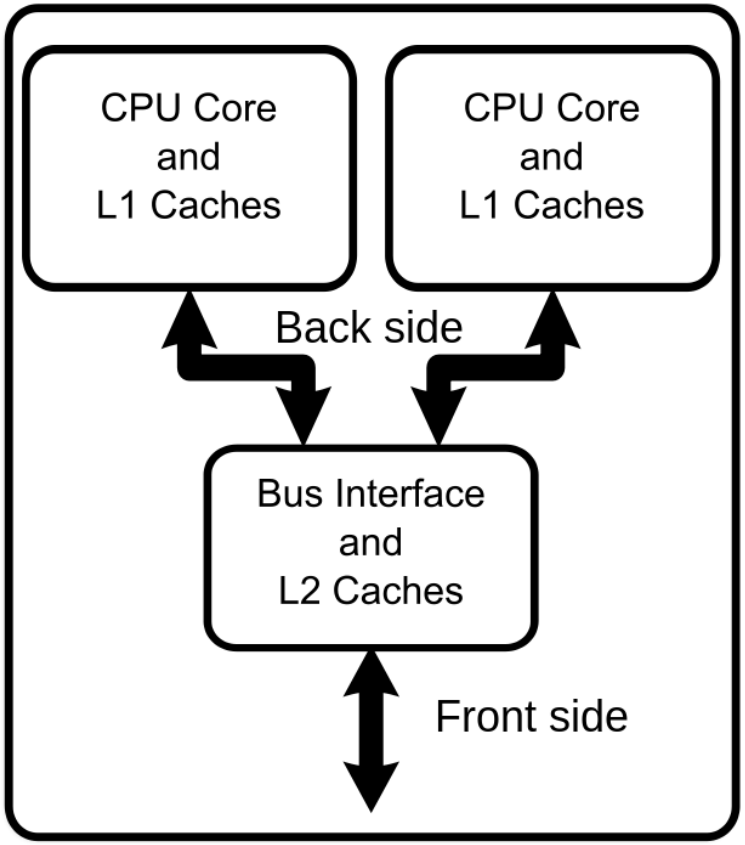
Programming Background — Hardware

CPU frequency (clock speed) growth is slowing



QuantEcon

Chip makers have responded by developing multi-core processors



Source: Wikipedia

QuantEcon

Exploiting multiple cores / threads is nontrivial

Sometimes we need to redesign algorithms

Sometimes we can use tools that automate exploitation of multiple cores

Hands On Exercises

Let's see vectorization, JIT compilation and parallelization in action

Please open

- `John/vectorization_numba.ipynb`

More information:

- The `Need for Speed` lecture on QuantEcon

QuantEcon

Distributed/Cloud Computing

Advantages: Can run computationally intensive code on big machines we don't own

Options:

- University machines
- AWS and other commercial services

QuantEcon